CENT System Documentation

Release 1.0

University of Helsinki

October 15, 2012

CONTENTS

CHAPTER

ONE

CENT APPLICATION - USER'S DOCUMENTATION

1.1 System Overview

1.1.1 What is CENT?

CENT (Computer Enabled Neuroplasticity Treatment) is an application for a novel treatment method for various brain diseases, esspecially Attention Deficit Hyperactivity Disorder (ADHD). CENT is monitoring patient's brain activity and using different visual stimulations, encouraging patients to alter their brain activities. This treatment is devoid of any side effects and has long-term effectiveness.

1.1.2 Installation

Note: CENT works on *Windows 7*, on both *32 bit* and *64 bit* architectures. Recommended is *Windows 7, 64 bit*.

Warning: Before following installation steps, make sure that your EEG Cap drivers are installed. More about that in *Supported EEG Devices* section.

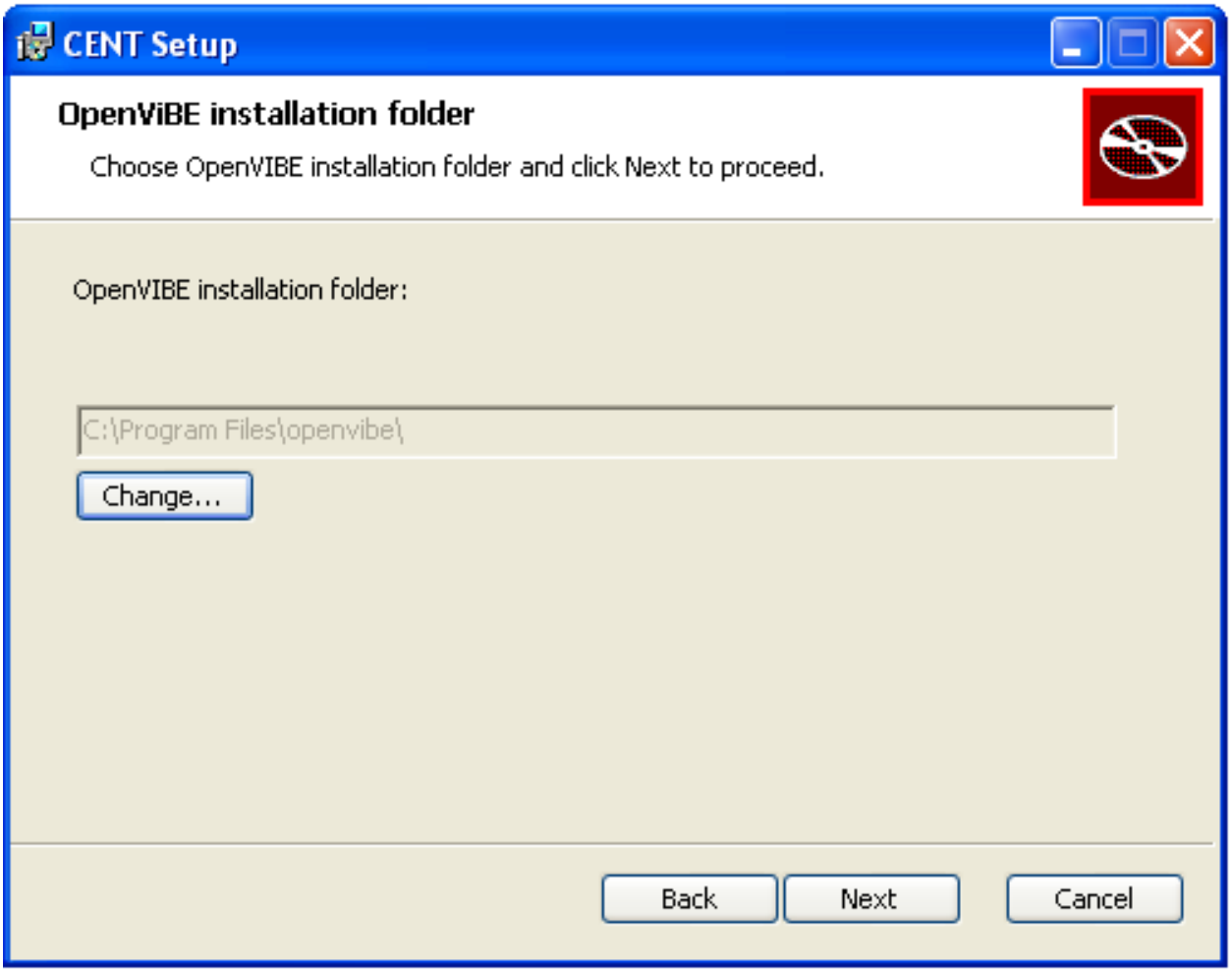
These instructions describe how to install **CENT Application**:

1. CENT uses OpenViBE platform. It has to be downloaded from http://openvibe.inria.fr/downloads and installed first (for documentation look at http://openvibe.inria.fr).

Warning: Remember location of the OpenViBE installation folder, you will need that information during CENT installation.

2. Install CENT Application from **CENTInstaller.msi**

CENT installer will guide you through the installation process. One important thing to underline is, that at some point you will be asked to indicate where OpenViBE is installed.

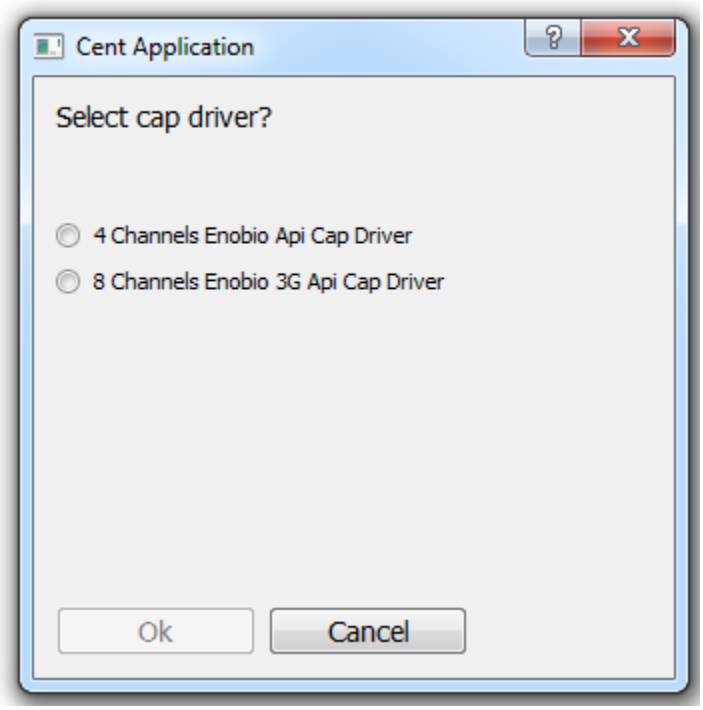


3. At this point you are ready to start CENT application.

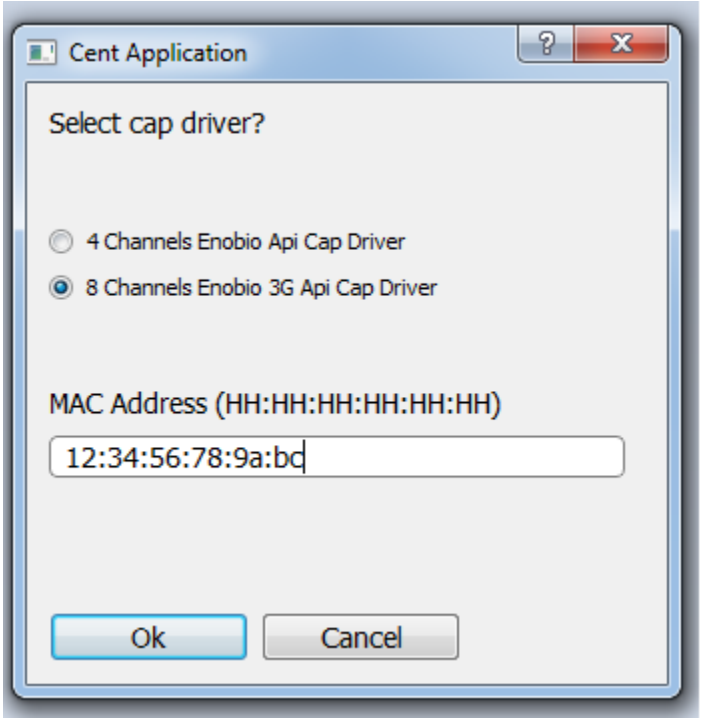
1.1.3 CENT Application Usage

Application Start

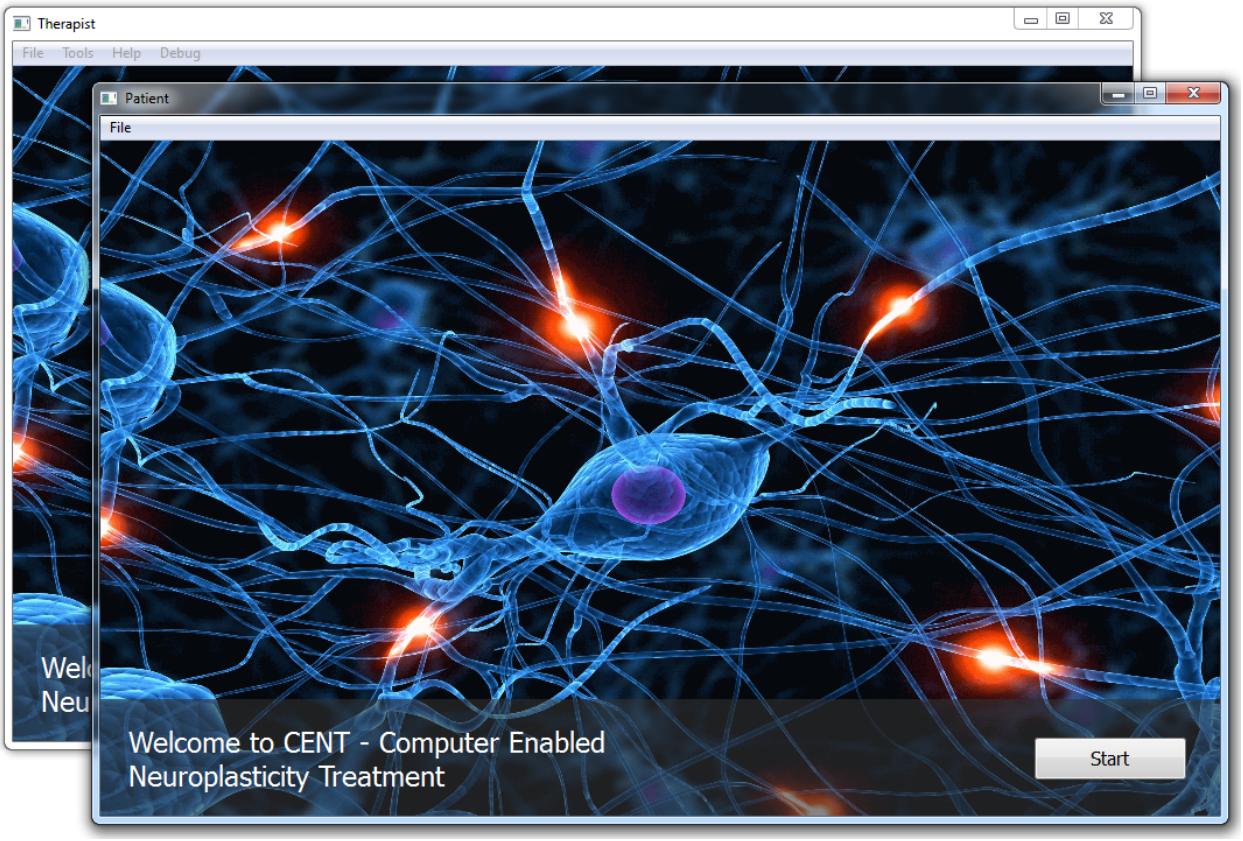
CENT application, when started, will present Select Cap Drivers window. The choice we have two cap drivers 4 and 8 channels.



For 8 channels cap driver we have to set MAC address needed to run (it's not possible go to next step without MAC address). MAC is store in memory.



Application is designed to work with two monitors - patient's and therapist's. CENT application, will present two similar windows - first on therapist's monitor, second - on patient's. Both windows have a name in the window header, so they can be recognized.



Those windows are not independent - any activities will affect both of them. For example "Start" button in patient's or therapist's window will switch to the next step on both windows.

Backup Settings

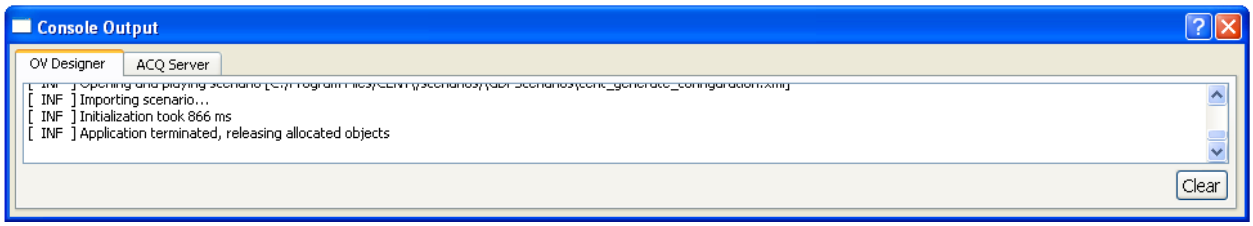
To choose backup location therapist have to open settings (*Tool->Settings*). In new window click "Change" and choose destination directory. This is a one-way synchronization of patient's data (whole *Patients* directory - `My Documents\CENT\Patients`). Backup is incremental, so locally deleted files won't be removed from the backup.

Backup under the hood

This information might be usefull if you happen to have trouble with the backup functionality. Underneath Cent uses `robocopy` cmd utility provided with Windows 7. The backup is invoked after every new file created. Failures are silent (only logs). The command used is: `ROBOCOPY [source] [destination] /S /LOG:robocopy.log`. It goes without saying that the user that runs Cent, has to have appropriate write access on the destination path.

OpenViBE Console Output

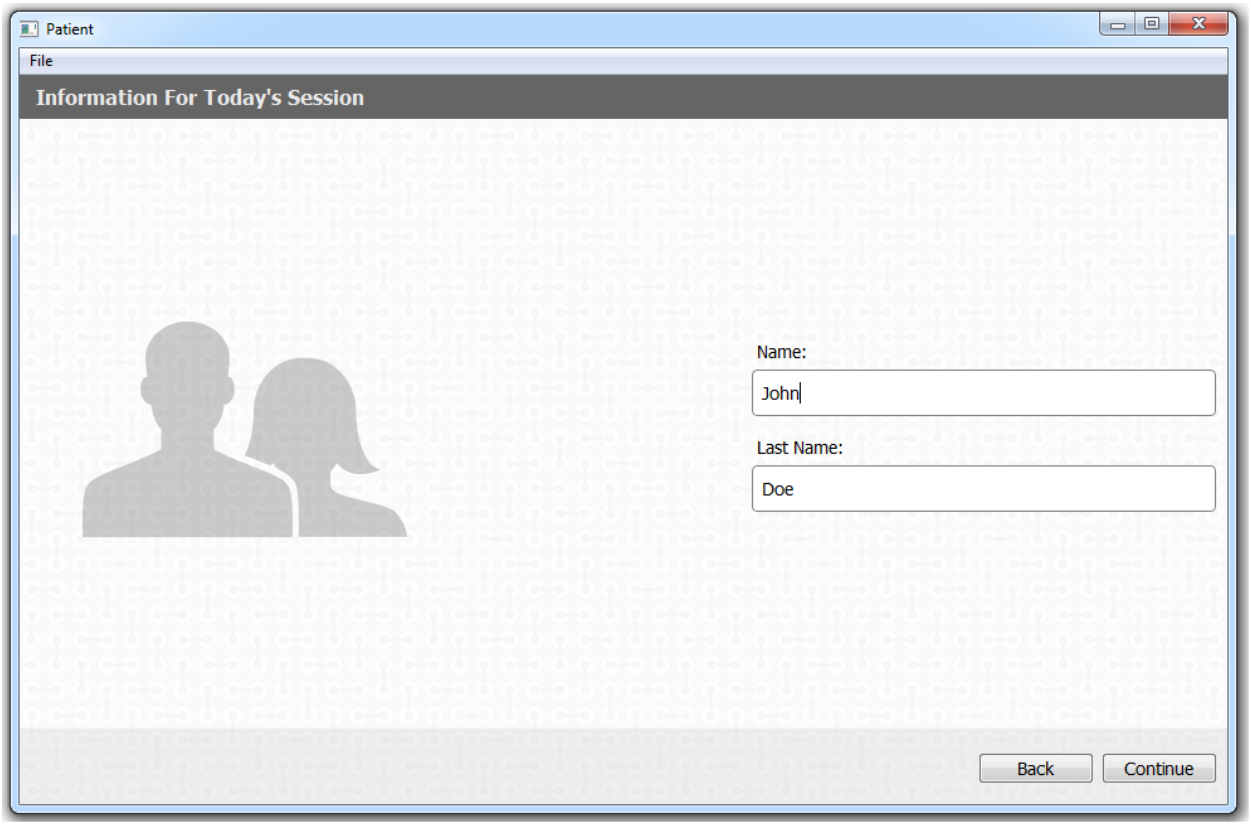
More advanced users of the CENT System, especially those creating new signal processing schemes, may benefit from the insight into the more detailed messages about the internal happenings. To open OpenViBE Console use *Debug->OpenViBE Console Output* in application menu. It will open new window with two tabs which will let you see messages comming from the Acquisition Server and from the OpenViBE Designer running in the background.



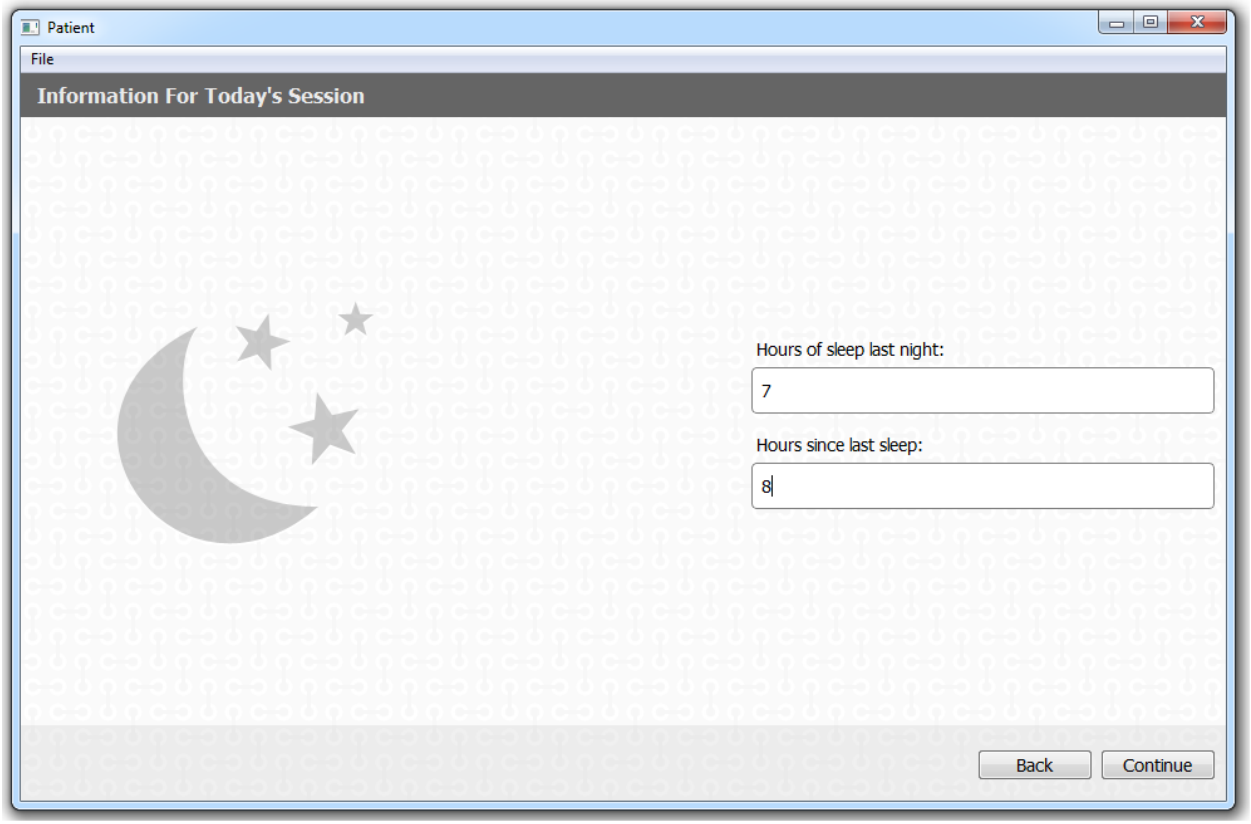
Patient Information Gathering

After clicking on "Start" there will be next page regarding patient's information. First patient type her/his name and last name. There are allowed letters, numbers and '_' character. Click "Continue" for next page. If it is new patient you have to choose IEP folder if it does not exist in patient directory. The IEP folder should contain:

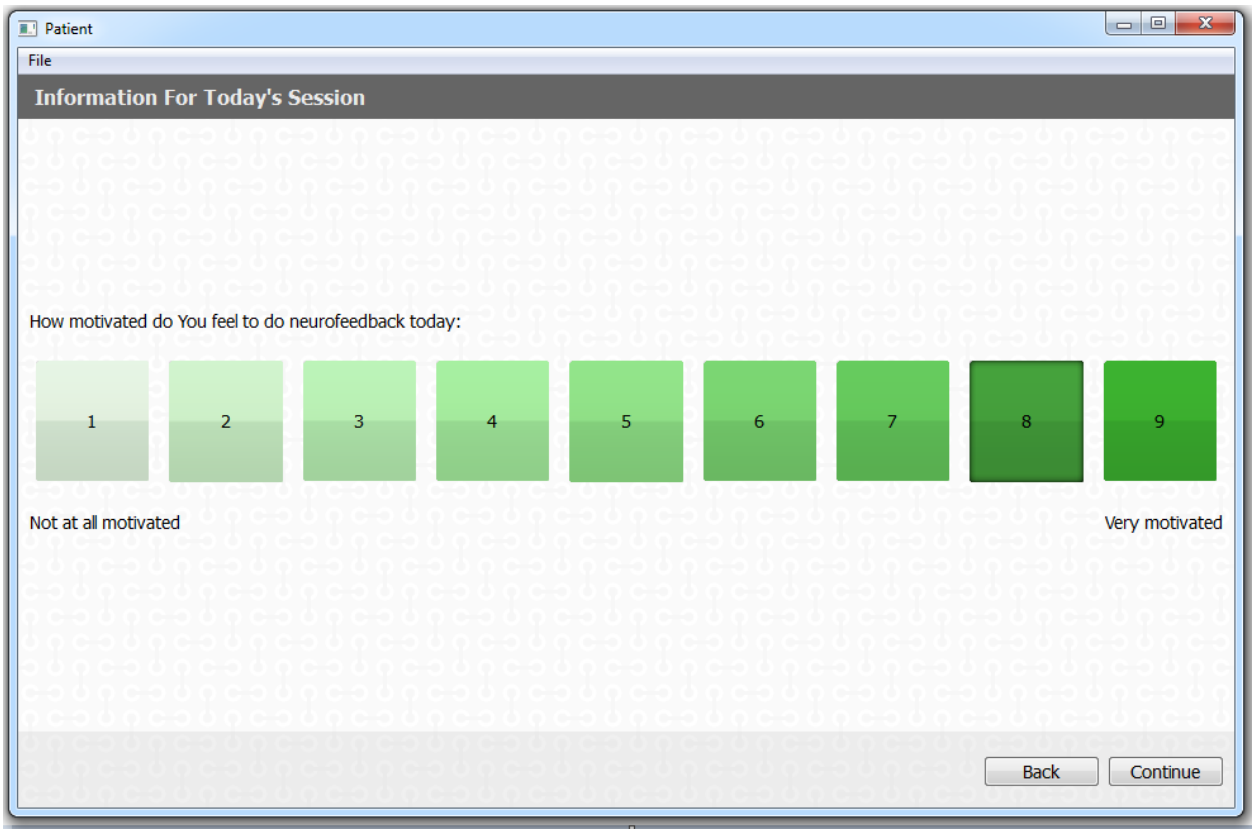
- `IEP\class_beta.cfg`
- `IEP\class_theta.cfg`
- `IEP\default_baseline\thetabase.cfg`
- `IEP\default_baseline\betabase.cfg`
- `IEP\default_baseline\baseline_spectrum.csv`



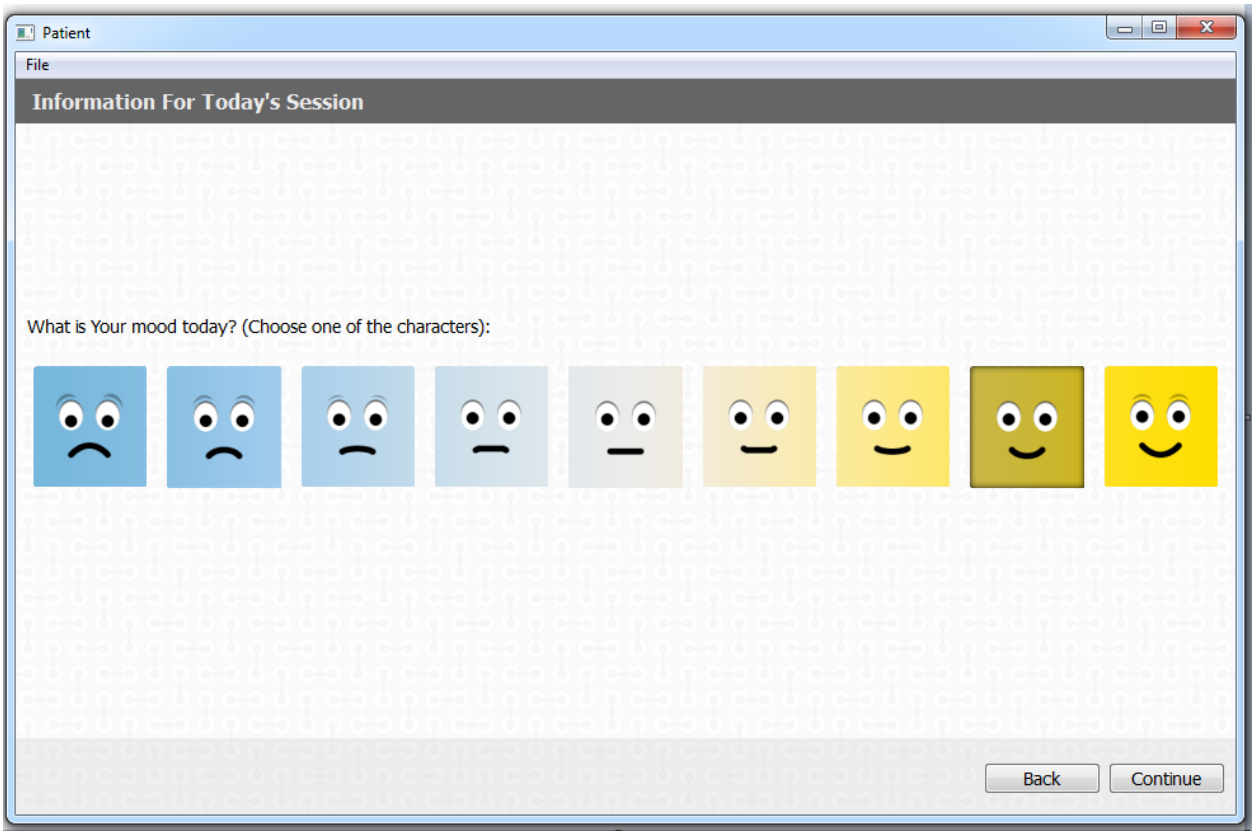
Next step is about the quality of the last night's sleep. Patient should provide his last night's sleep duration (max 24) and information about how much time elapsed since he woke up (maximum 72). Click "Continue" for next page or get back to patient's name screen with "Back" button.



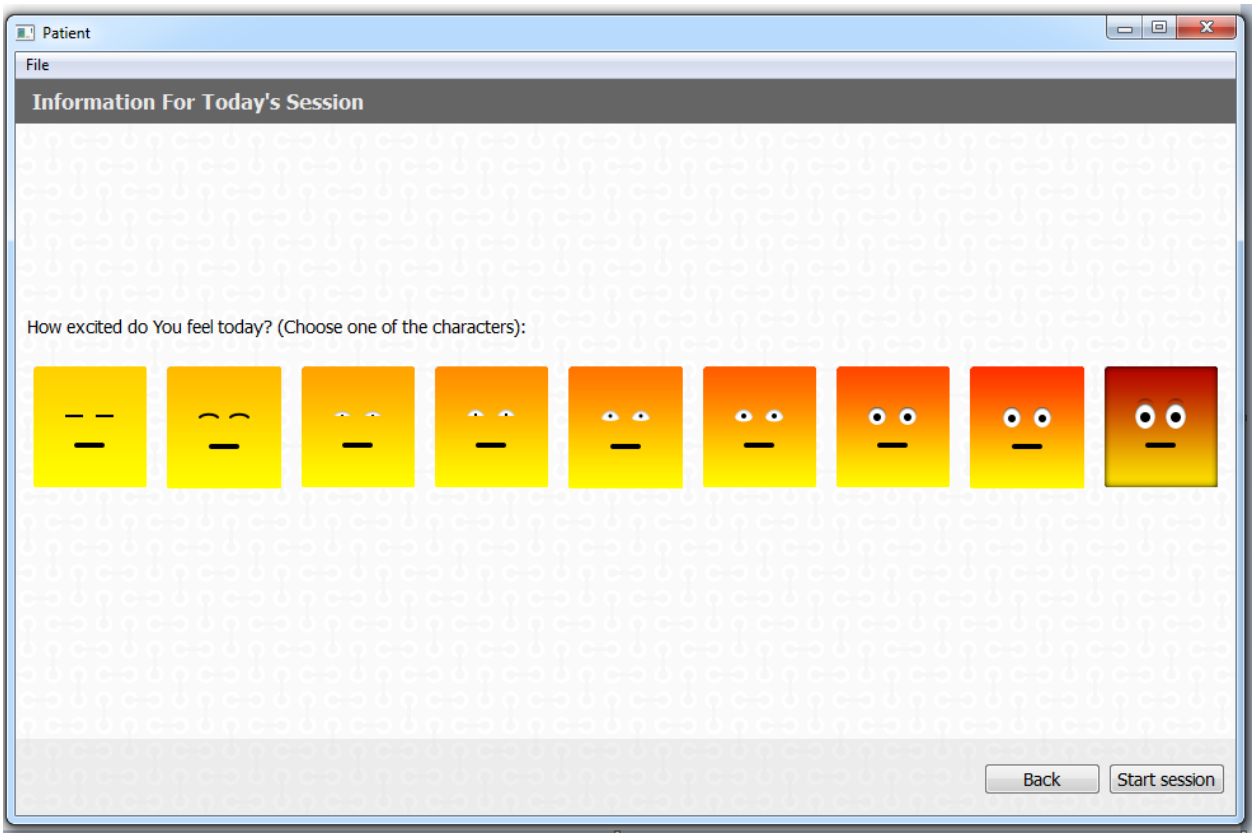
Motivation plays an important factor. In the next screen patient needs to ask himself how motivated to do the session she/he feels.



Patient's mood follows - patient have to choose one of nine characters:



And the last thing is to choose how excited patient feels today - patient have to choose one of nine characters:



At this point, patient or therapist can start the session with "Start session" button.

Note: Information about the patient - hours of sleep, mood, excitement, etc. are saved in a text file:

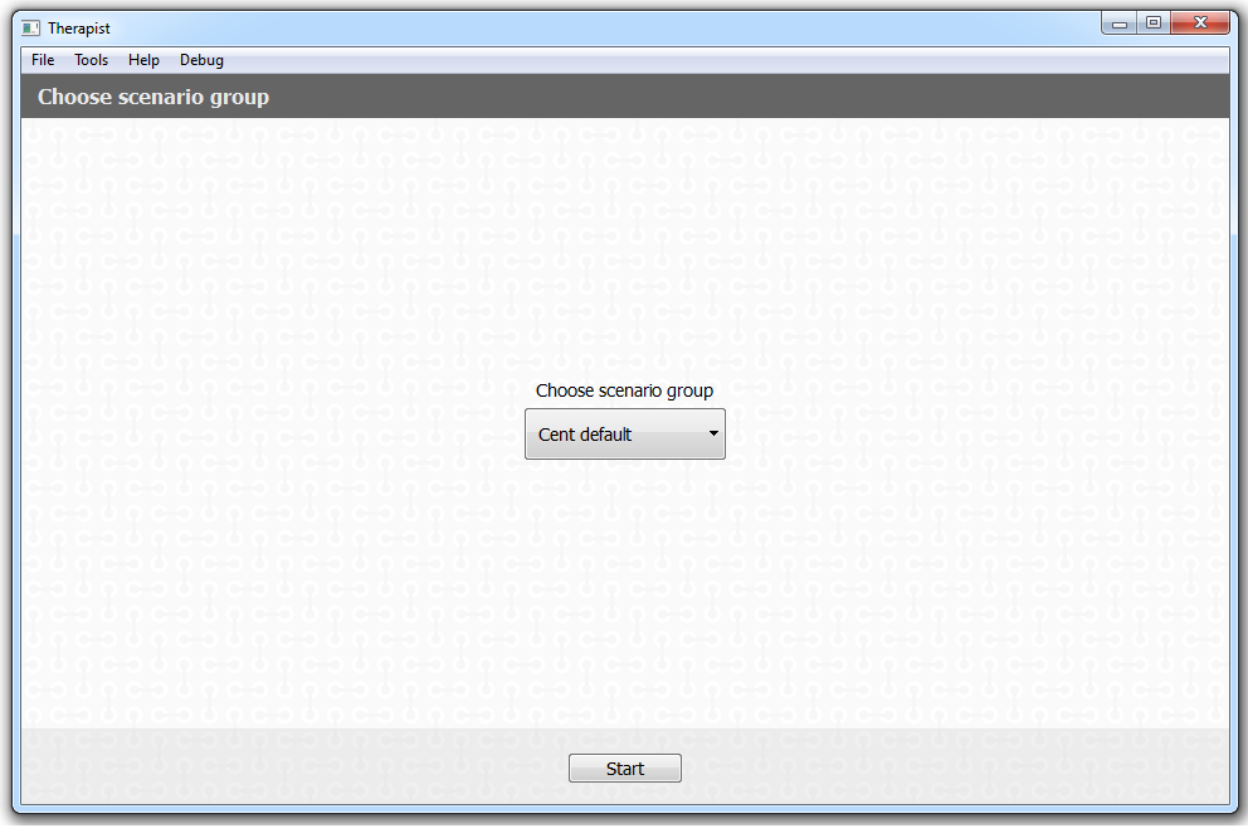
`My Documents\CENT\Patients\[OBFUSCATED_PATIENT_NAME]\session_[DATESTAMP]\patient_condition`

For more information about patient data see *Patient data storage*.

Note: Patient's name is obfuscated to secure his privacy. There is a separate tool (*Mapping Tool*) provided, which is able to deobfuscate those names. This application is available only to designated people.

OpenViBE Scenario Group selection

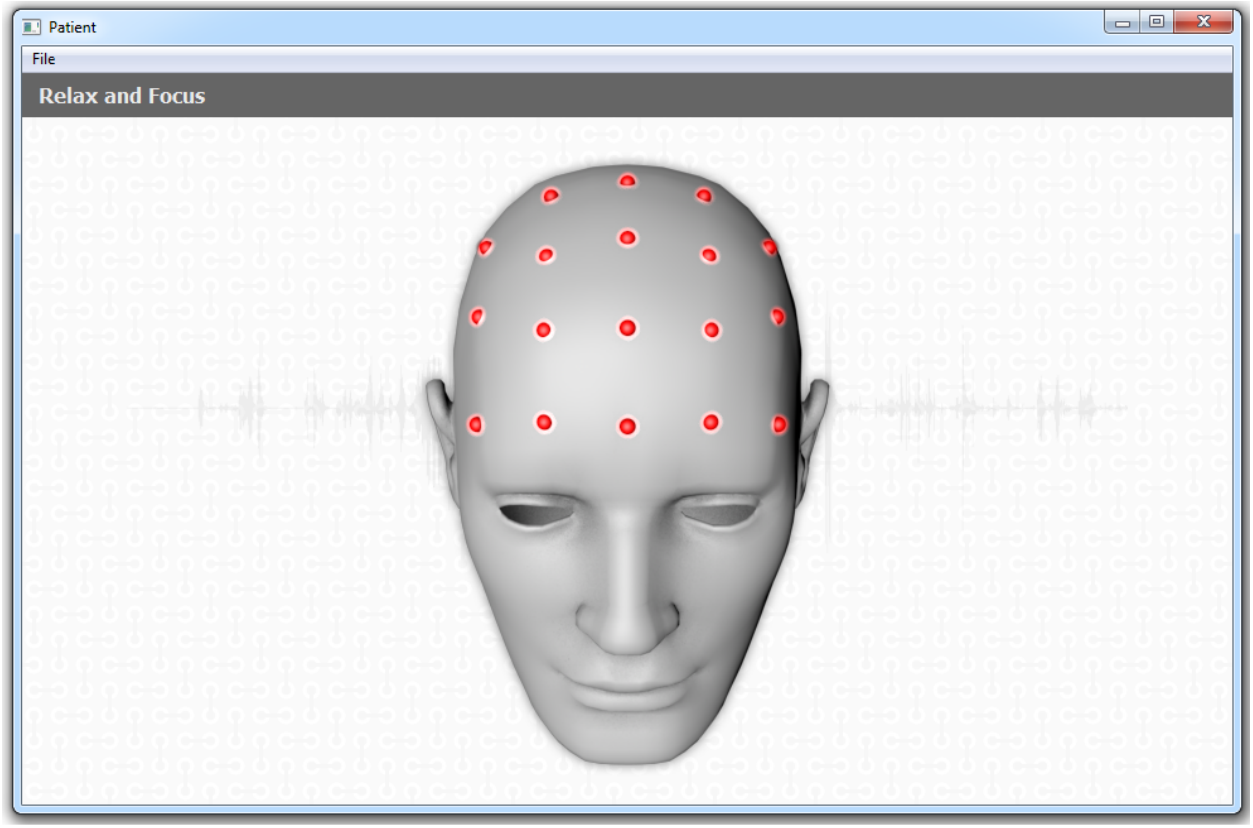
On the next page, therapist must choose scenario group. CENT application is using OpenViBE scenario files to process EEG signals. Different scenarios may be in use, depending on the used EEG cap, and other factors. To add new scenario group you have to create new directory in `[CENT_INSTALLED_FOLDER]\scenarios`. Directory name will be used as a name of the scenario group.



EEG Cap Setup

Note: This information is for Enobio® device but other devices could be similar

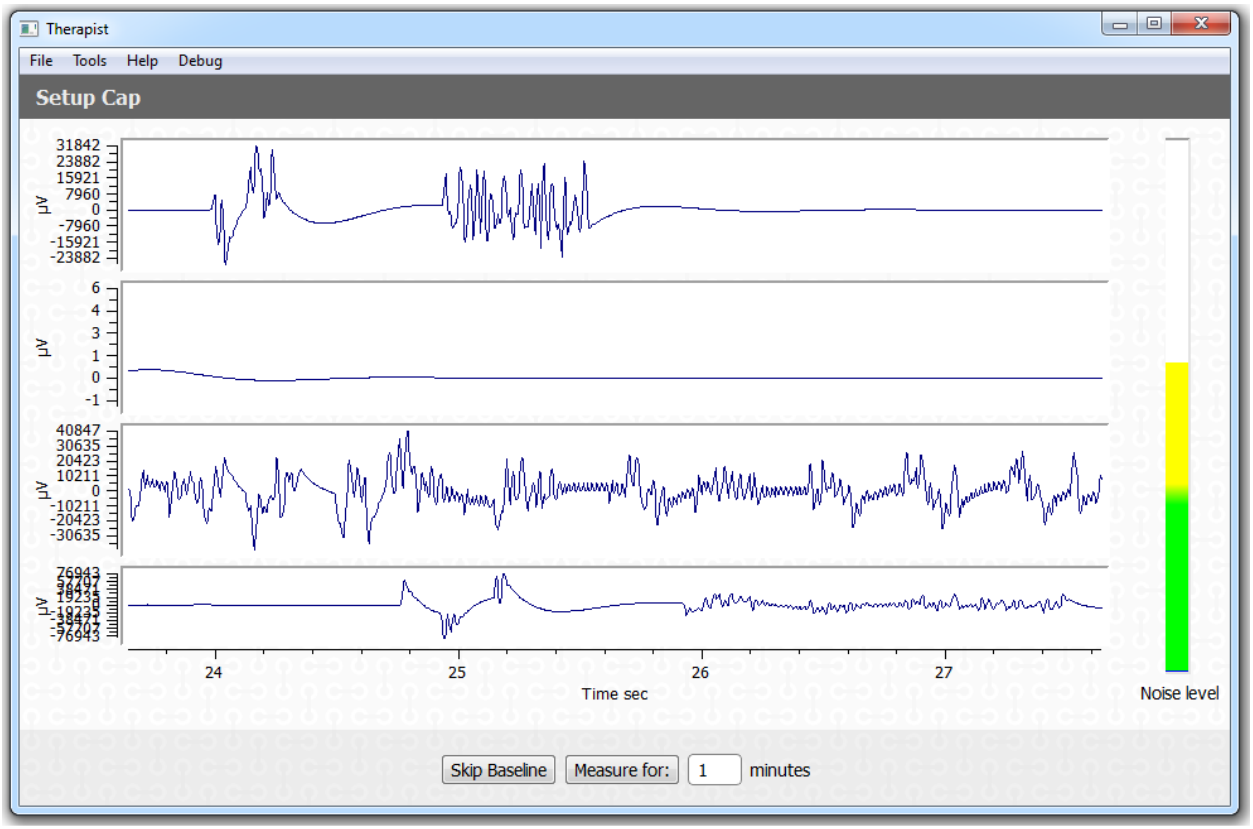
Patient have to put Enobio® cap on his/her head. Then therapist attach the Enobio® to the cap using the velcro. The next thing therapist place holders in the cap holes, insert the gel and plug the electrodes. Therapist can also put dry electrodes on the headband. There is also one electrode for a right ear. The last thing to do is to plug the USB reciver into the computer. More information you find in device manual.



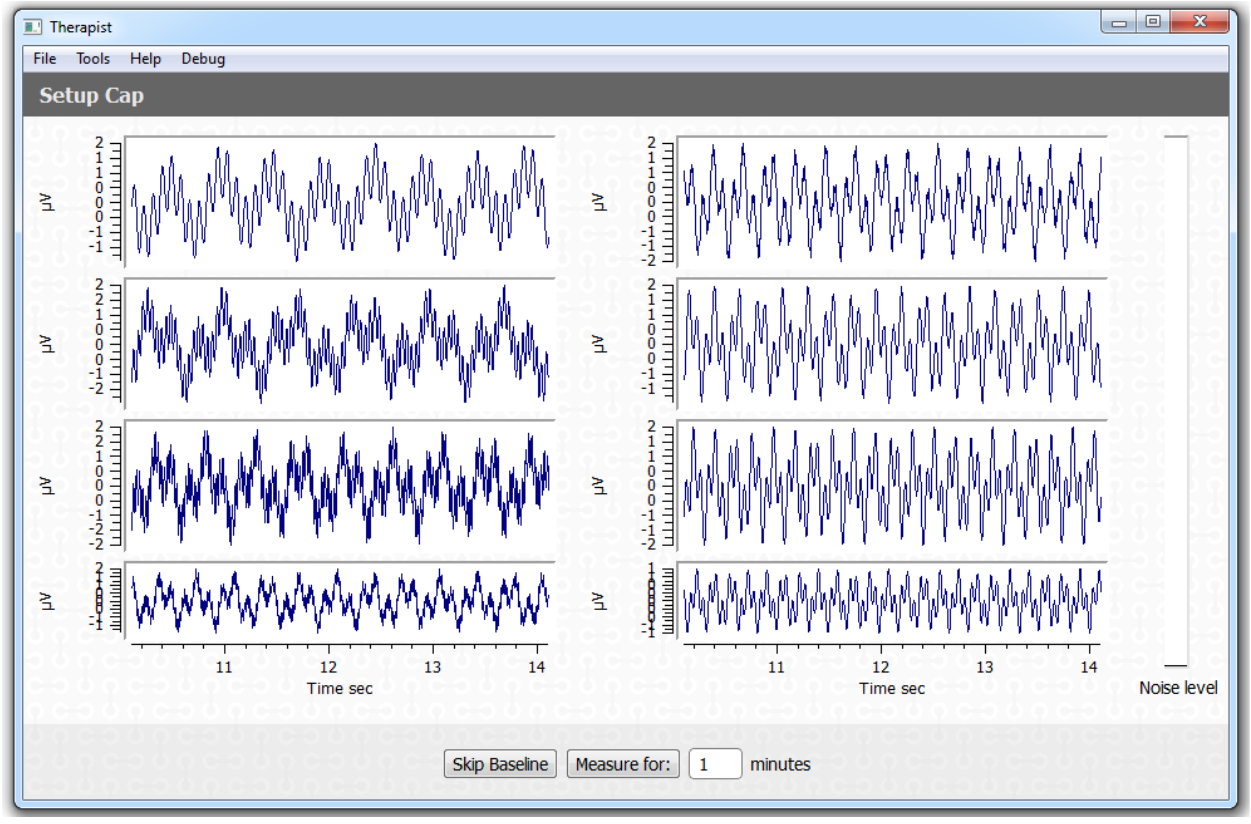
Baseline Measurement

When EEG Cap is correctly placed and connected, therapist can proceed to the baseline measurement. Therapist must enter measurement time and ask patient to look at the cross in the center of the window.

Note: Therapist should observe noise level indicatior to be able to decide if measurement should be taken again.



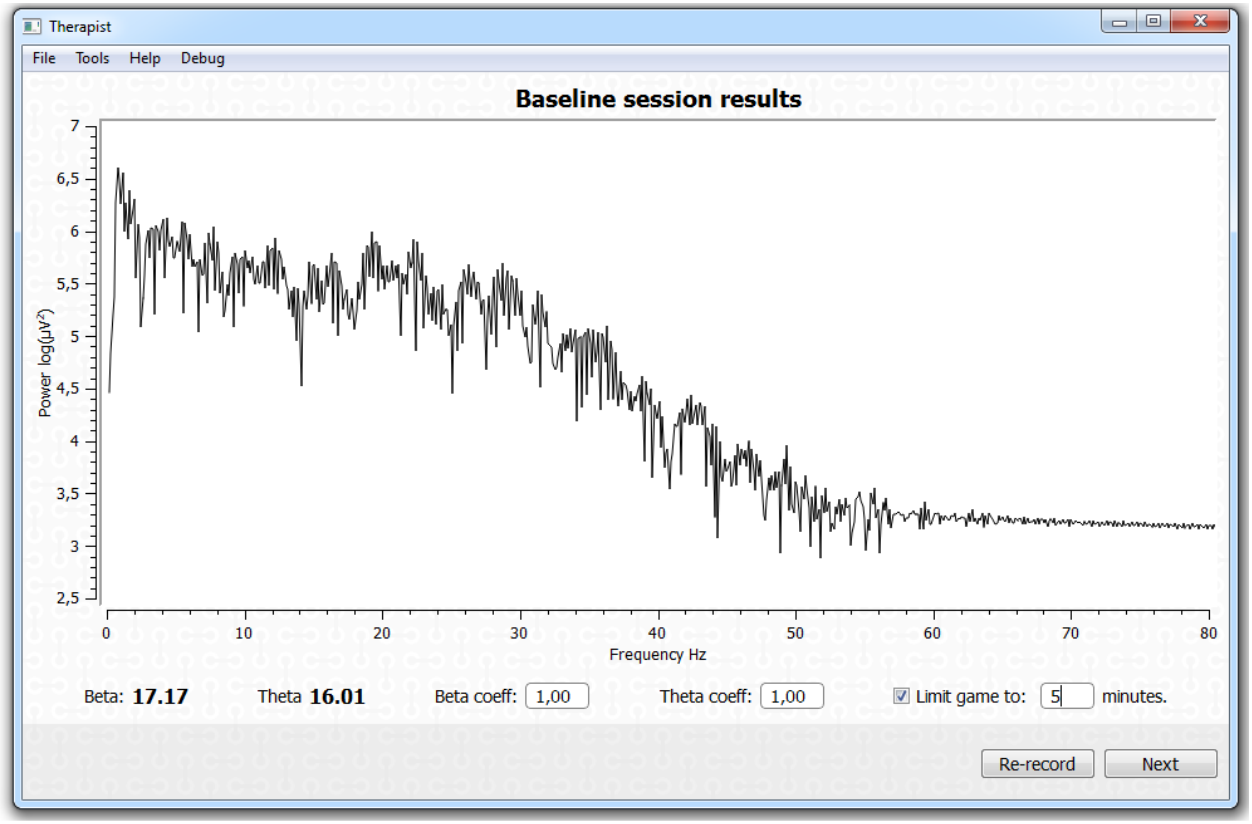
4 channels noise level.



8 channels noise level.

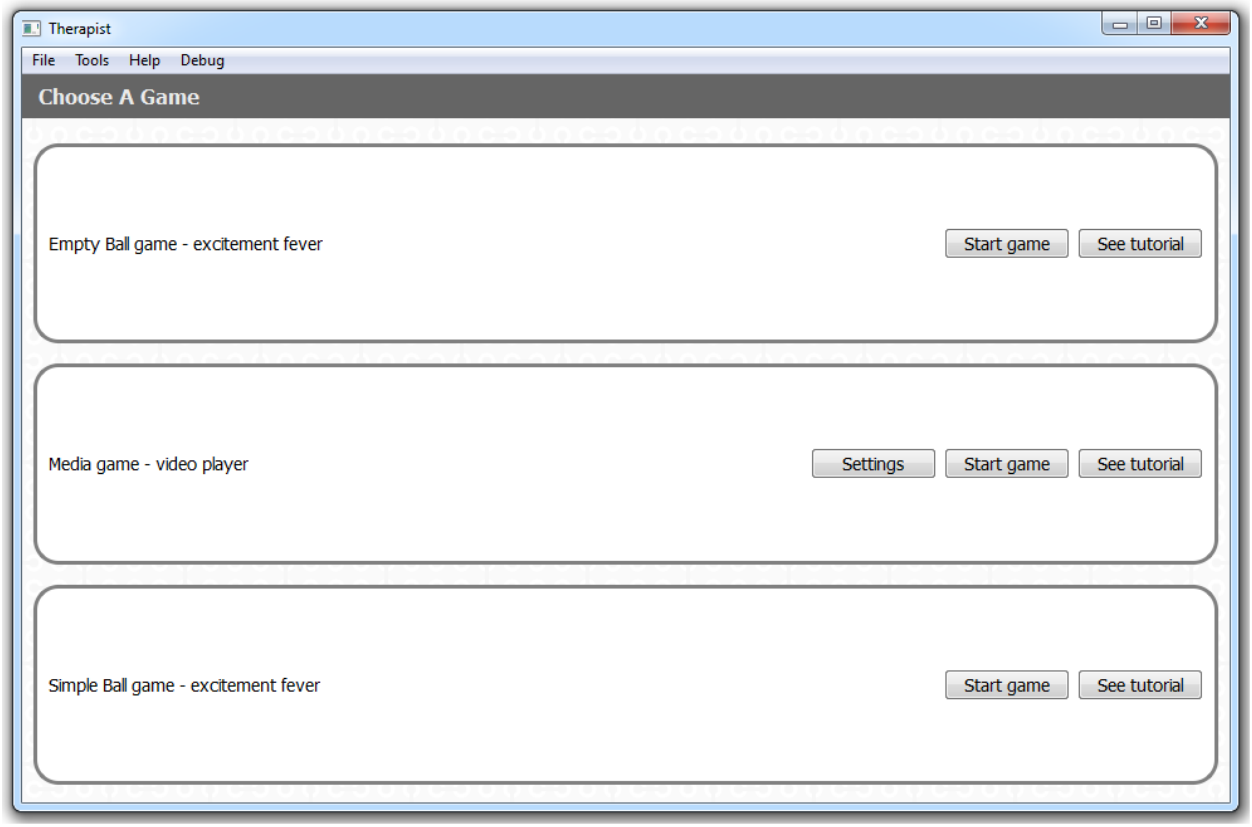
Therapist can end baseline measurement earlier than specified using “End recording” button.

After baseline measurment, therapist will be presented with the EEG signal power spectrum and calculated Beta and Theta power values. At this point therapist should enter coefficients for both of those power bands, as well as information about duration of the game that will follow.



Game Selection

After baseline measurement, CENT is expecting patient or therapist to choose a game which will be played during the session. List of available games will be presented in both windows. Each game has a tutorial which explains the rules of the game.

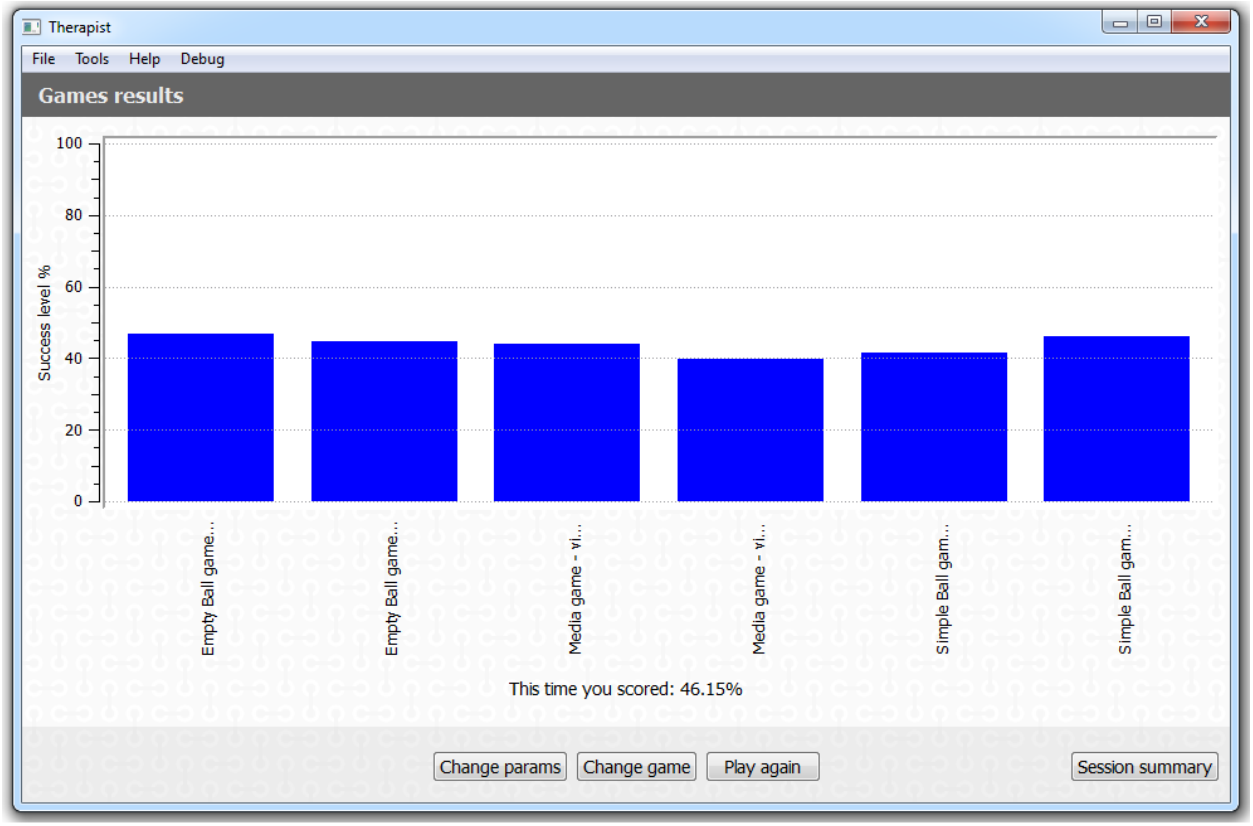


Gameplay

During the game therapist can observe EEG chart on his window. Patient should focus on the game. Game will end after pressing "End game" button on therapist window or after some time which was set by therapist.

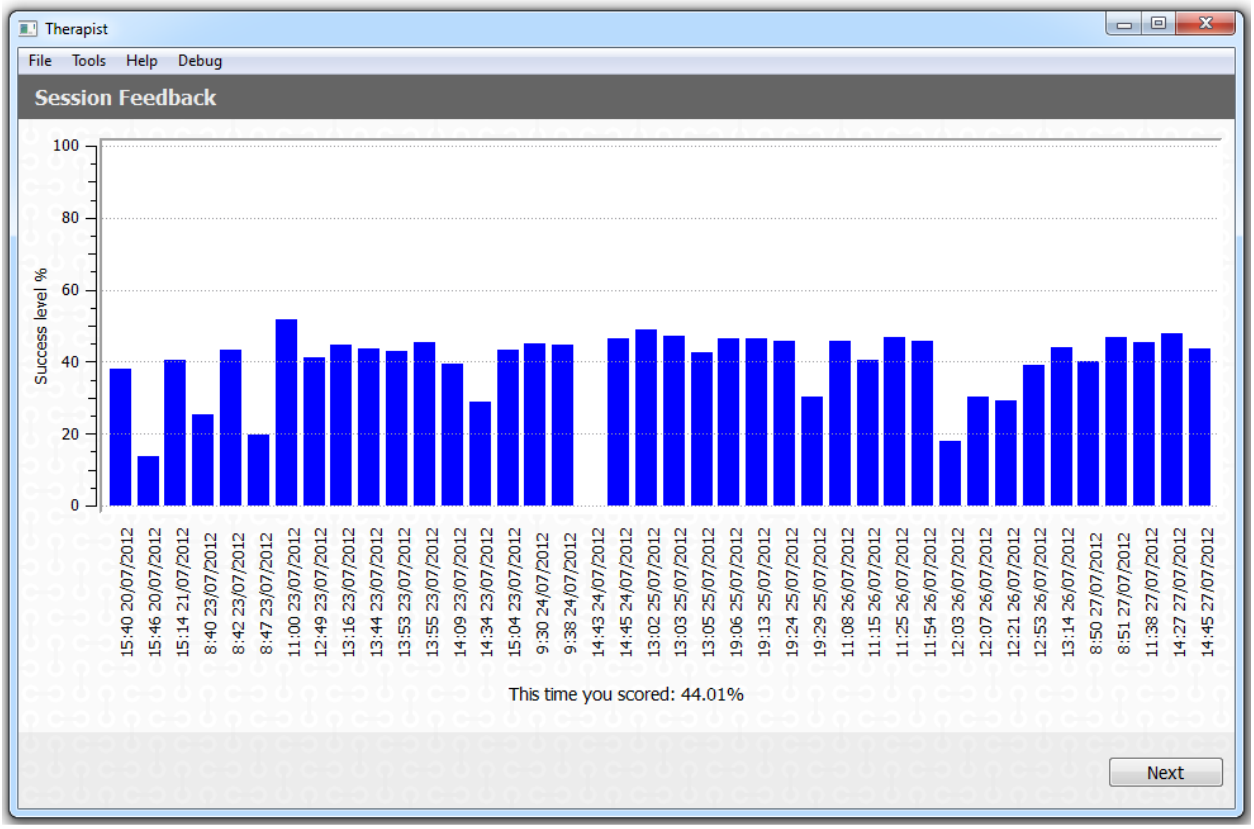
Game Summary

After the game result charts will be presented in both windows. Each bar on the chart represents percentage of the time spent in positive classification (desired state of mind). Patient can play again or change the game. Therapist can also get back to the baseline results to change parameters or show session summary.



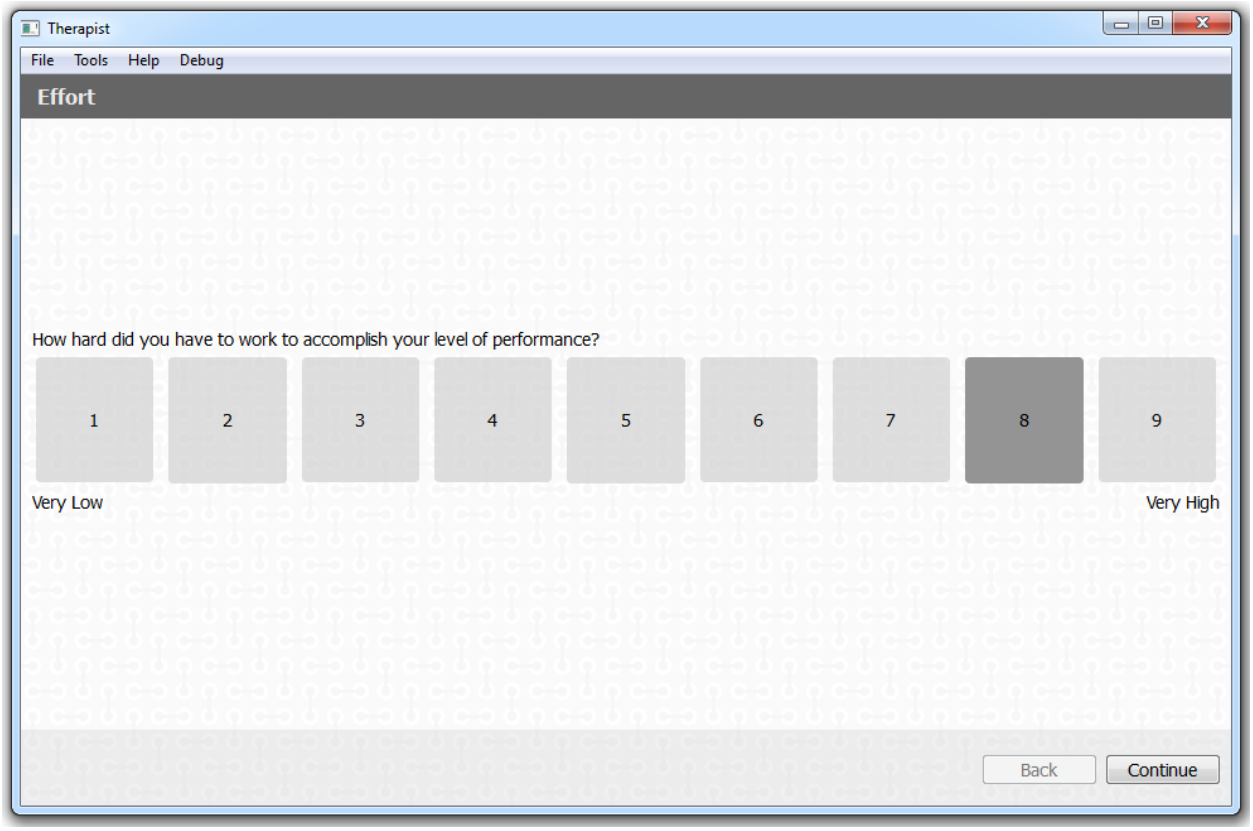
Session Summary

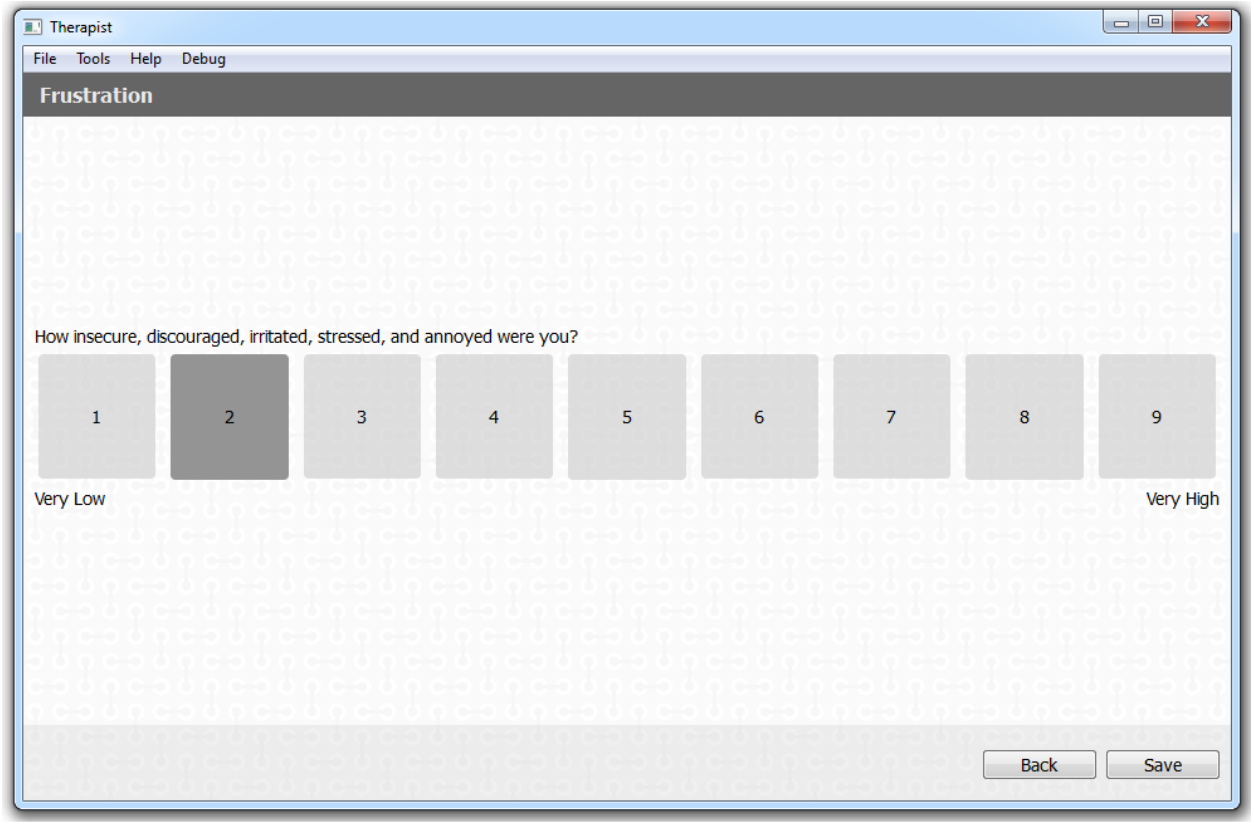
Whole session consisting of number of the games is summarized in similar way. On this chart, each bar represents percentage of the time spent in positive classification (desired state of mind) during the sessions (one bar for all the games playd within one session).



Patient's Evaluation

After the session patient is asked two follow up questions.


Therapist
File Tools Help Debug
Effort
How hard did you have to work to accomplish your level of performance?
1 2 3 4 5 6 7 8 9
Very Low
Very High
Back
Continue


Therapist
File Tools Help Debug
Frustration
How insecure, discouraged, irritated, stressed, and annoyed were you?
1 2 3 4 5 6 7 8 9
Very Low
Very High
Back
Save

End Session

Session may be ended with “End session” button. After that CENT application will present first screen, starting whole process from the beginning.

1.1.4 Mapping Tool

Mapping Tool is a command line application which creates mapping between hashes and real patient’s names. To use it, one shall execute it in the command line console, indicating localization of the folder with patients data:

```
MappingTool.exe K:\\PatientsFolder
```

Application will create a text file in current directory. File will follow simple structure:

```
[patient name];[encrypted patient name]
```

You will find MappingTool in *MappingToolPackage.zip*. It is accompanied by *QtCore4.dll*, which is required to run this application.

1.2 OpenViBE as a Digital Signal Processor in CENT

1.2.1 How OpenViBE is used by CENT System?

CENT application uses OpenViBE platform to process data from the EEG device. OpenViBE runs in the background with preselected scenarios. CENT starts OpenViBE when needed - while processing data from EEG device, during baseline measurement and when patient plays the game. There are different scenarios used for the baseline measurement and for the gameplay.

1.2.2 Protocols Development

Environment Variables

- "CENT_CURRENT_PATIENT"; - variable to Patient's folder: `My Documents\CENT\Patients\[CUR_PAT]`
- "CENT_CURRENT_SESSION"; - variable to Patient's current session folder: `My Documents\CENT\Patients\CUR_PAT]\session_[DATESTAMP]`
- "CENT_CURRENT_GAME"; - variable to Patient's current game folder: `My Documents\CENT\Patients\CUR_PAT]\session_[DATESTAMP]\games\[GAME_NAME]_[DATESTAMP]`
- "CENT_SCENARIOS_HOME"; - variable to CENT scenarios: `[CENT_INSTALLED_FOLDER]\scenarios\[SCENARIO_`
- "OPENVIBE_HOME"; - variable to OpenViBE install folder

Available Scenarios

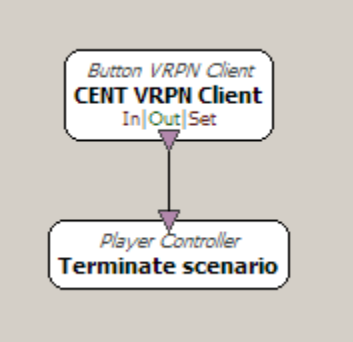
All scenarios used by CENT application are stored in `[CENT_INSTALLED_FOLDER]\scenarios\[SCENARIO_GROUP_NAME]`.

Every group scenario has to have these four scenarios:

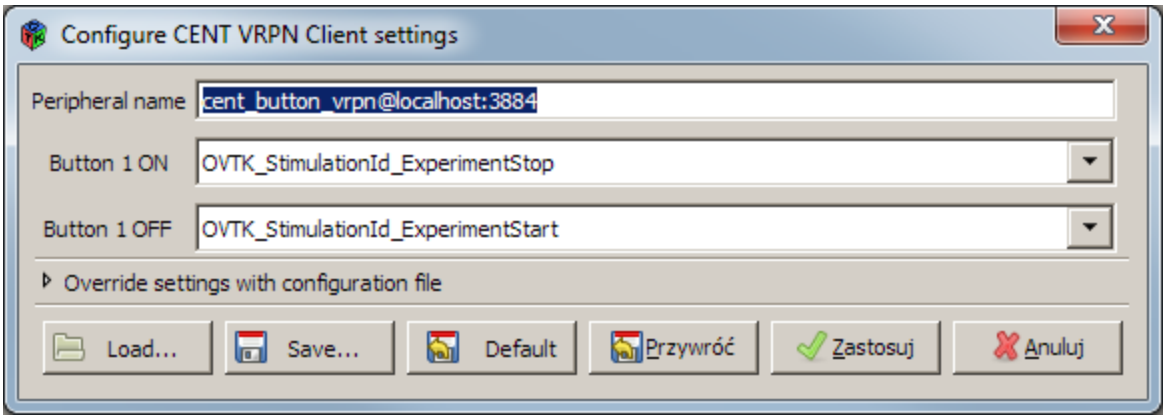
1. cent_monitoring_and_noise.xml
2. cent_baseline.xml
3. cent_generate_configuration.xml
4. cent_game.xml

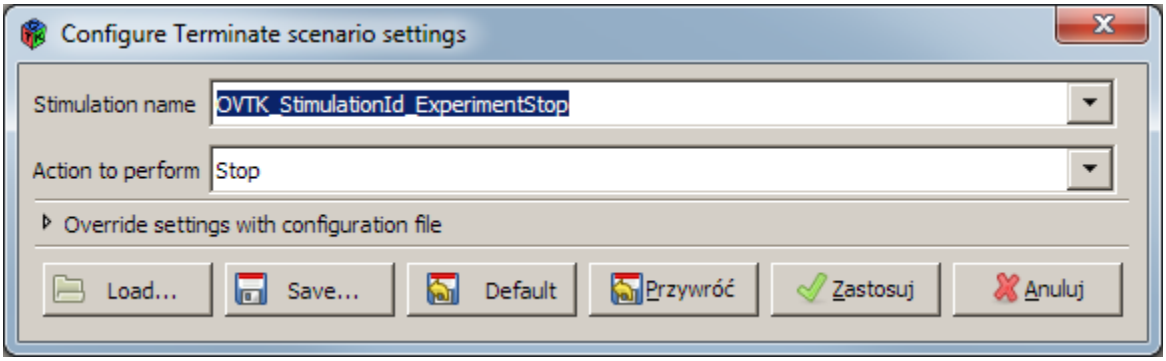
Scenarios requirements

Every scenario needs two boxes for the Cent to be able to stop it cleanly.



Their configuration is as follows:





Monitoring and noise scenario

This scenario is expected to provide live EEG signal and live noise level information, both through VRPN Analog Servers. Noise

1. Current noise value
2. Noise threshold1
3. Noise threshold2
4. Min value
5. Max value

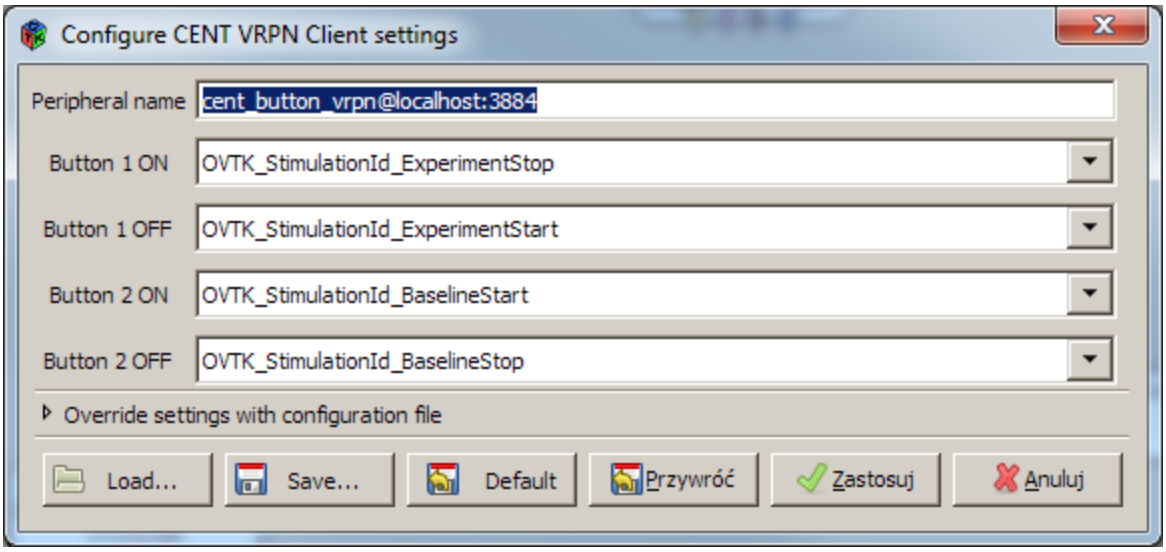
The servers are called accordingly: **liveEEG**, **CENTNoiseLevel**

Most scenarios will usually use Aquisition client to get EEG data, but it is not a requirement.

Baseline scenario

It is supposed to create a `[{CENT_CURRENT_SESSION}]\baseline_spectrum.csv` spectrum file and two intermediate files used by next scenario. These file are `[{CENT_CURRENT_SESSION}]\betaval.csv` and `[{CENT_CURRENT_SESSION}]\thetaval.csv`. It is also expected to provide live EEG signal and live noise level information, like the previous one.

Here the CENT VRPN Client is slightly extended to the base one. Additional stimulations can be used in stream switch box.

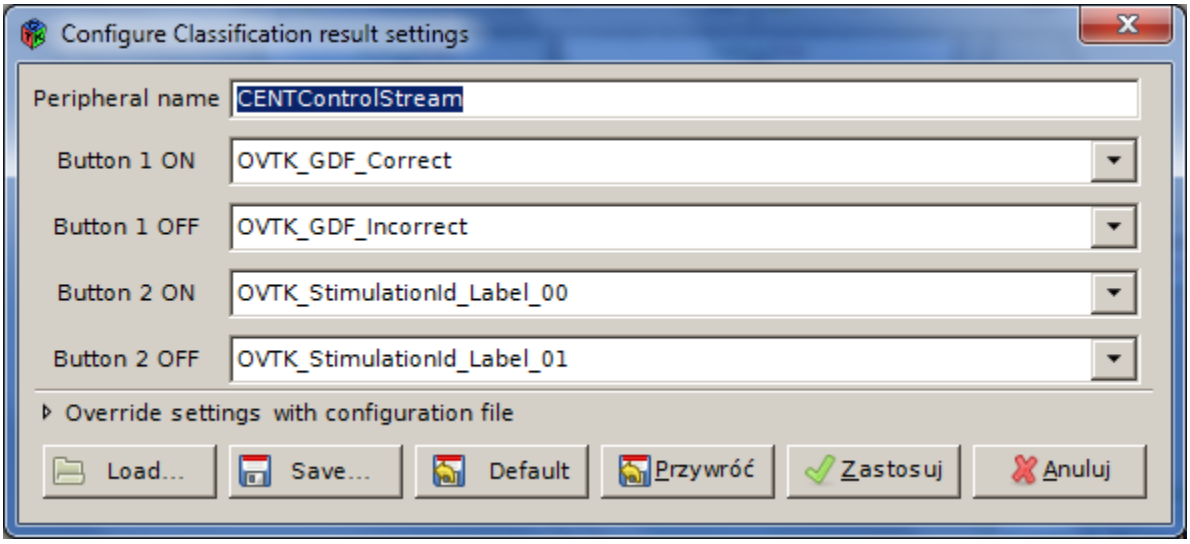


Generate configuration scenario

This one has to create two files used by next scenario. These file are `[{CENT_CURRENT_SESSION}]\betabase.cfg` and `[{CENT_CURRENT_SESSION}]\thetabase.cfg`.

Game scenario

Main functionality of this scenario is to provide classification outcome and epoch notificatio through VRPN Button Server configured as follows:



The epoch notification is like a beacon informing about new classification to overcome the limitation of VRPN Button Server that is notifying the client only on value change. One way is to do it via Lua script like that:

```
1 dofile(os.getenv("OPENVIBE_HOME") .. "/share/openvibe-plugins/stimulation/lua-stimulator-stim-codes.l
2
3 stim = OVTK_StimulationId_Label_00
4 -- this function is called when the box is initialized
```

```
function initialize(box)
        io.write("initialize has been called\n");

        -- inspects the box topology
--        io.write(string.format("box has %i input(s)\n", box:get_input_count()))
--        io.write(string.format("box has %i output(s)\n", box:get_output_count()))
--        io.write(string.format("box has %i setting(s)\n", box:get_setting_count()))
--        for i = 1, box:get_setting_count() do
--                io.write(string.format(" - setting %i has value [%s]\n", i, box:get_setting(i)))
--        end

end

-- this function is called when the box is uninitialized
function uninitialize(box)
        io.write("uninitialize has been called\n")
end

-- this function is called once by the box
function process(box)
        io.write("process has been called\n")

        -- enters infinite loop
        -- cpu will be released with a call to sleep
        -- at the end of the loop
        while true do

                -- gets current simulated time
                t = box:get_current_time()

                if box:get_stimulation_count(1)==1 then -- stimulation received
                        box:send_stimulation(1,stim,t,0)
                        if stim == OVTK_StimulationId_Label_00 then
                                stim = OVTK_StimulationId_Label_01
                        else
                                stim = OVTK_StimulationId_Label_00
                        end
                        box:remove_stimulation(1, 1)
                end
                -- releases cpu
                box:sleep()
        end
end
```

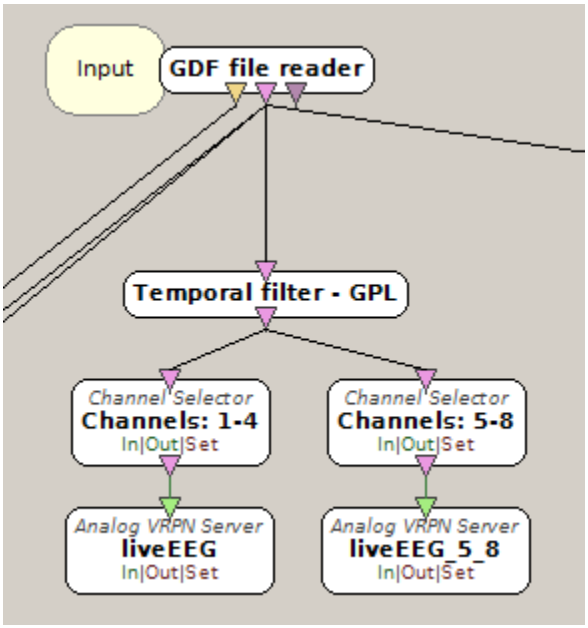
Another important thing is to publish Current theta/beta powers through 4 Analog channels of VRPN server called CENTInGa

1. theta value
2. theta extent
3. beta value
4. beta extent

It is also expected to provide live EEG signal and live noise level information, like the previous ones.

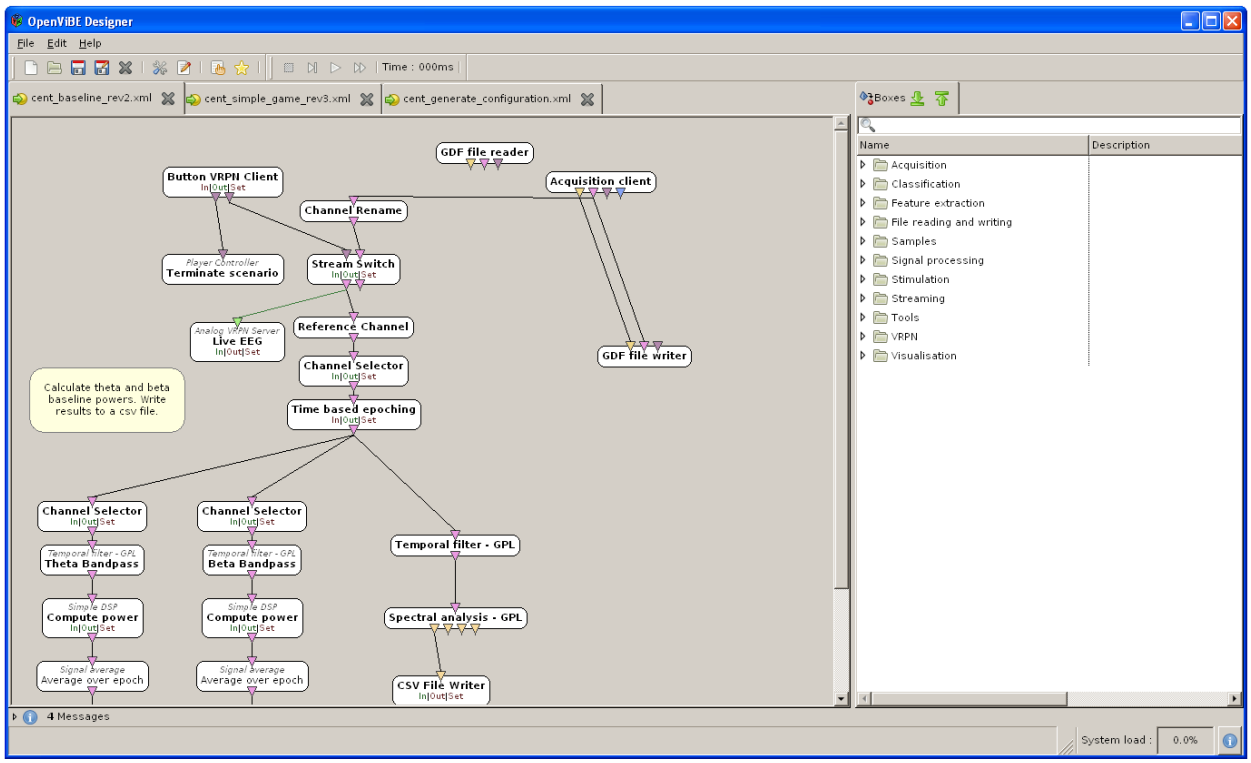
8 channel scenarios

8 channel scenarios has to forward live EEG channels from 5 to 8 via additional VRPN Analog Server called **liveEEG_5_8** like that:



Adding scenario groups

You can create new scenarios in OpenViBE designer. Just create **.xml* file with scenario and modify it. More information about creating simple scenario you find on official OpenViBE site at: http://openvibe.inria.fr/documentation-index/#User+Documentation



1.3 Supported EEG Devices

1.3.1 Starlab® Enobio

General Information

Enobio® is a wearable, modular and wireless electro-physiology sensor system for the recording of EEG, ECG and EOG. Enobio® has 4 channels, bandwidth from 0 to 125 Hz and wireless (IEEE 802.15.4) communication. More information about Enobio® you find in manual or on http://starlab.es/products/enobio

Installation

Note: This information are for Enobio® 2.0

Plug in the USB disk from Enobio® to the computer. You find there Enobio® 2.0 User Manual and Enobio® 2.0 Quick Start Guide. Plug in the USB port wireless receiver.

If the drivers will not install automatically. Install it from Device Manager (My Computer -> Properties -> Device Manager). Find new device on the list and update driver choosing folder *usb_drivers* on Enobio® USB disk.

Enobio® device now is ready to work with CENT application.

1.3.2 Other EEG Devices

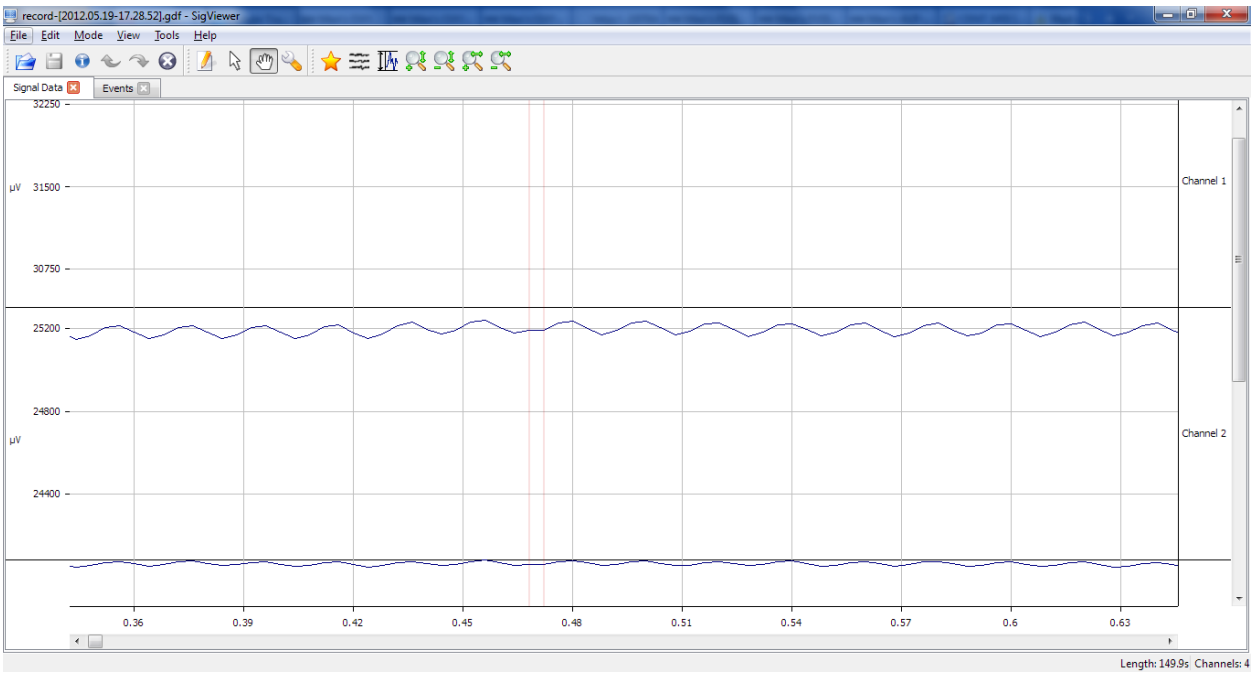
CENT Application is able to use different EEG Devices, as long as those are supported by OpenViBE platform. In other case, new drivers can be developped. See brief overview of the drivers development in *EEG Cap Drivers Development* section.

1.4 How Acquisition Server collect samples

Happy scenario is that the cap provides 250 samples per second times four channels and all the Acquisition Server has to do is to push it forward to Designer. Yeah that almost never happens. There are many reasons from computer being stuck at something (even for 100ms is bad) to many kinds of noise (environmental or due to patient fault, like skin movement).

When something bad happens with the sample, Enobio cap flags that sample as "LOST PACKET". Our driver ignores those packets.

In the Acquisition Server that situation is called a drift (it can happen both ways: too many or too little data). To guarantee 250Hz it adds interpolated samples to the sending buffer. The best thing about it, is that it flags samples also with stimulations: OVTK_GDF_Correct and OVTK_GDF_Incorrect. You can see them in SigViewer when you open any GDF file written by CENT.



There are this pink vertical lines (stimulations) and the first stands for Incorrect and the second Correct one. What is between them is interpolated by Acquisition Server, so nothing random.

Conclusion is that if you want to be super accurate you can take those stimulations into account when processing signal in OV Designer. I don't think you have to worry about that but wanted you to know a little bit about the under the hood stuff.

1.5 Patient data storage

1.5.1 General information

All patient data is stored under `C:\Users\[username]\Documents\CENT\Patients`. Under every patients' obfuscated folder are folders for every session. Session folder name has an embedded date and time. For example `session_20120519183421` Is a session recorded on 19/05/2012 at 18:34.21. Next to the sessions' folders is an IEP folder put there by CENT on patient creation. The therapist has to provide a valid original IEP folder for this functionality to work.

1.5.2 Privacy security and it's limitations

Every patient's folder name is obfuscated for privacy reasons. However there are some limitations to that. When thinking about privacy of patient's information one needs to be aware that a clever user with access to CENT system and it's folder structure can still figure out which folders belong to which patients.

Simplest way to go around this protection is to click "save as" option in notepad when it has `session_notes.txt` open for current patient. Another approach would be to check timestamps of the session folders. Knowing when a patient recorded his or hers session it's easy to guess which folder is who's.

The solution (obfuscated folder name) is a good approach covering basic privacy rules. It's architecture is based on known usage patterns for CENT system. Authors of the software do not take responsibility for an incorrect or improper use of the software.

CHAPTER

TWO

CENT APPLICATION - DEVELOPER'S DOCUMENTATION

2.1 CENT ZIP Packages

We provide two zip packages:

- CENTPackage.zip
- CENTSources.zip

In CENTPackages are: Installer, Mapping Tool, Documentation and GameTutorial In CENTSources are all sources to build CENT application.

To build Cent system you will also need 3rd party packages. They are widely available as open source programs on their websites (with information on the version we used):

- http://qwt.sourceforge.net/ (version 6.0.1)
- http://www.cs.unc.edu/Research/vrpn/ (version 07.29)
- http://openvibe.inria.fr/ (version 0.12.0-svn3107)

For unit tests you will also need visual leak detector:

- http://vld.codeplex.com/ (version 2.1.0)

2.2 Development Environment Setup

2.2.1 Required Software

- Qt 4.8 SDK
- Microsoft Visual Studio 2010
- OpenViBE software source codes
- Qwt sources

2.2.2 Required System Variables

- **CENT_SDK** - directory where You have checked out the SDK for CENT

- **VRPN** - path to the VRPN code
- Add Qt binaries folder to the **PATH** system variable

2.3 Build Instructions

2.3.1 Code Checkout

Checkout from GIT

Note: Those links work for BLStream employees, you should ask for correct ones some of the university's representatives.

- `git+ssh://[ACRONYM]@git.blstream.net/project/cent-test.git`
- `git+ssh://[ACRONYM]@git.blstream.net/project/cent_games.git`
- `git+ssh://[ACRONYM]@git.blstream.net/project/cent_openvibe.git`
- `git+ssh://[ACRONYM]@git.blstream.net/project/cent_qwt.git`
- `git+ssh://[ACRONYM]@git.blstream.net/project/cent_sdk.git`
- `git+ssh://[ACRONYM]@git.blstream.net/project/cent_vrpn.git`

Getting the code as an archive

Not having access to the git.blstream.net, you can get source code as a ZIP archive, which is a part of the final delivery, and at disposal of the University's CENT team representatives.

2.3.2 Build CENT Qt Application

- First you need to build VRPN from the directory that **VRPN** environment variable is pointing to.
- Import the **.pro* file into Visual Studio:

```
qmake -r -tp vc %CENT_SDK%/CENTApplication/CENTApplication.pro
```

- Open the generated solution file and build.
- Before run remember to put *qwt.dll* into output directory of CENT system

2.3.3 Build OpenViBE on Windows

- Download and put the sources of the latest OpenVibe into:

```
%CENT_SDK%\OpenVibe\
```

- Run and wait for install:

```
%CENT_SDK%\OpenVibe\scripts\win32-install_dependencies.exe
```

- Copy:

```
%CENT_SDK%\OpenVibe\scripts\win32-init_env_command.cmd-skeleton
```

- Rename it to:

  ```
  %CENT_SDK%\OpenVibe\scripts\win32-init_env_command.cmd
  ```

- Run in command line:

  ```
  %CENT_SDK%\OpenVibe\scripts\win32-init_env_command.cmd
  ```

  ```
  %CENT_SDK%\OpenVibe\scripts\win32-build.cmd
  ```

- OpenViBE will be built in:

  ```
  %CENT_SDK%\OpenVibe\dist\
  ```

2.3.4 Build an OpenViBE Box

You should build OpenViBE with the box code inside the plugins folder. The dll with the plugin will be put into:

```
%CENT_SDK%\OpenVibe\dist\share\openvibe-plugins\[your-plugin-path]
```

To optimize the build process please refer to the OpenViBE documentation

To see the OpenViBE box in the designer create an *openvibe.conf* file in `C:\Users\[user]` and fill with:

```
Designer_ShowUnstable = true
```

2.3.5 Build OpenViBE Acquisition Server with Starlab® Enobio® Driver

Go to:

```
%CENT_SDK%\OpenVibe\openvibe-applications\acquisition-server
```

Then issue command:

```
mklink /D /J cent ..\..\..\..\EnobioAcquisitionServer\AcquisitionServer
```

Edit previously created *win32-init_env_command.cmd* and set CENT as a branch to be built. Appropriate line should look like:

```
SET OpenViBE_application_acquisition_server_branch=cent
```

Rebuild as usual. You may find it handy to run:

```
%CENT_SDK%\OpenVibe\scripts\win32-generate-vc-proj.cmd
```

Which will configure Visual Studio project and drop it to:

```
%CENT_SDK%\OpenVibe\local-tmp/visual
```

2.3.6 Build Qwt

Download Qwt sources and put them in `%CENT_SDK%/SDK/qwt` Import Qt pro file, build, copy *qwt.dll* to CENT system output dir (don't change the **.pro* it's LGPL)

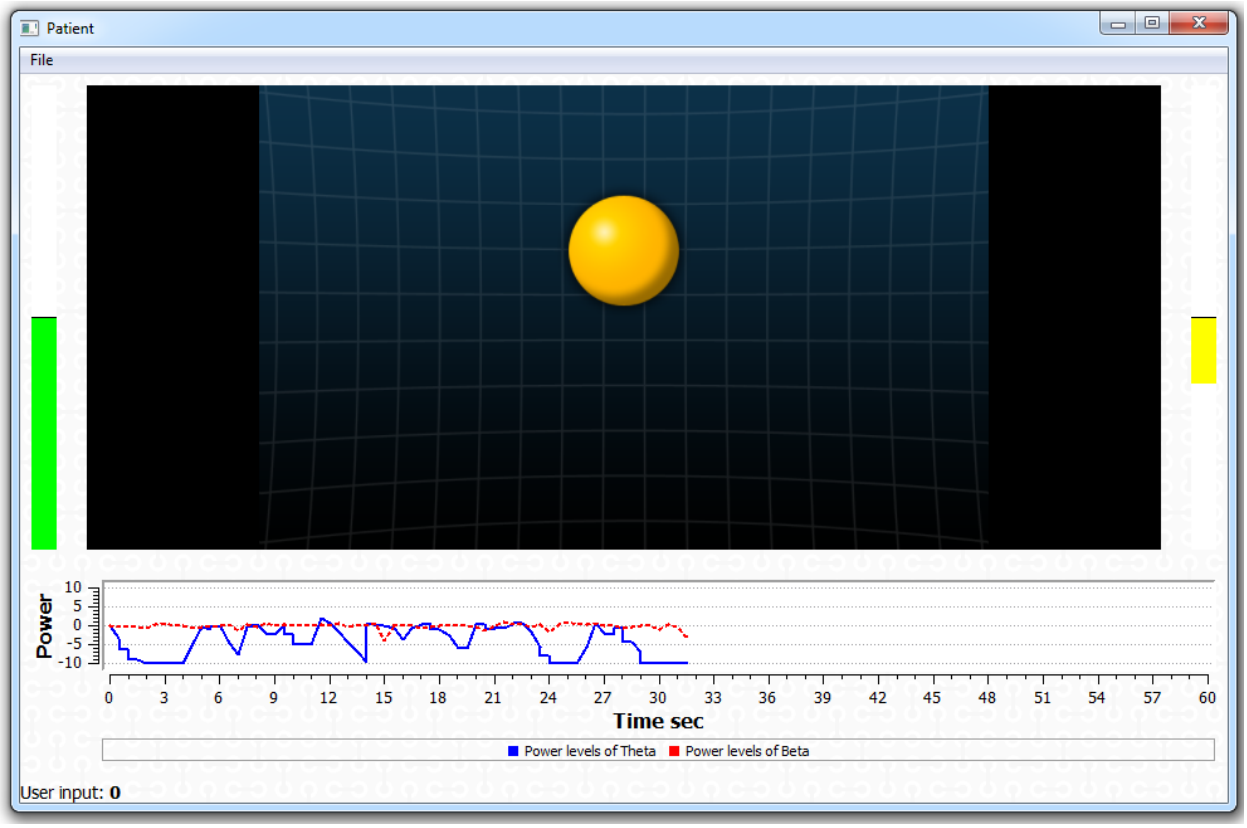
2.4 CENT Games Development

2.4.1 Currently Available Games

Currently are available three games in CENT application:

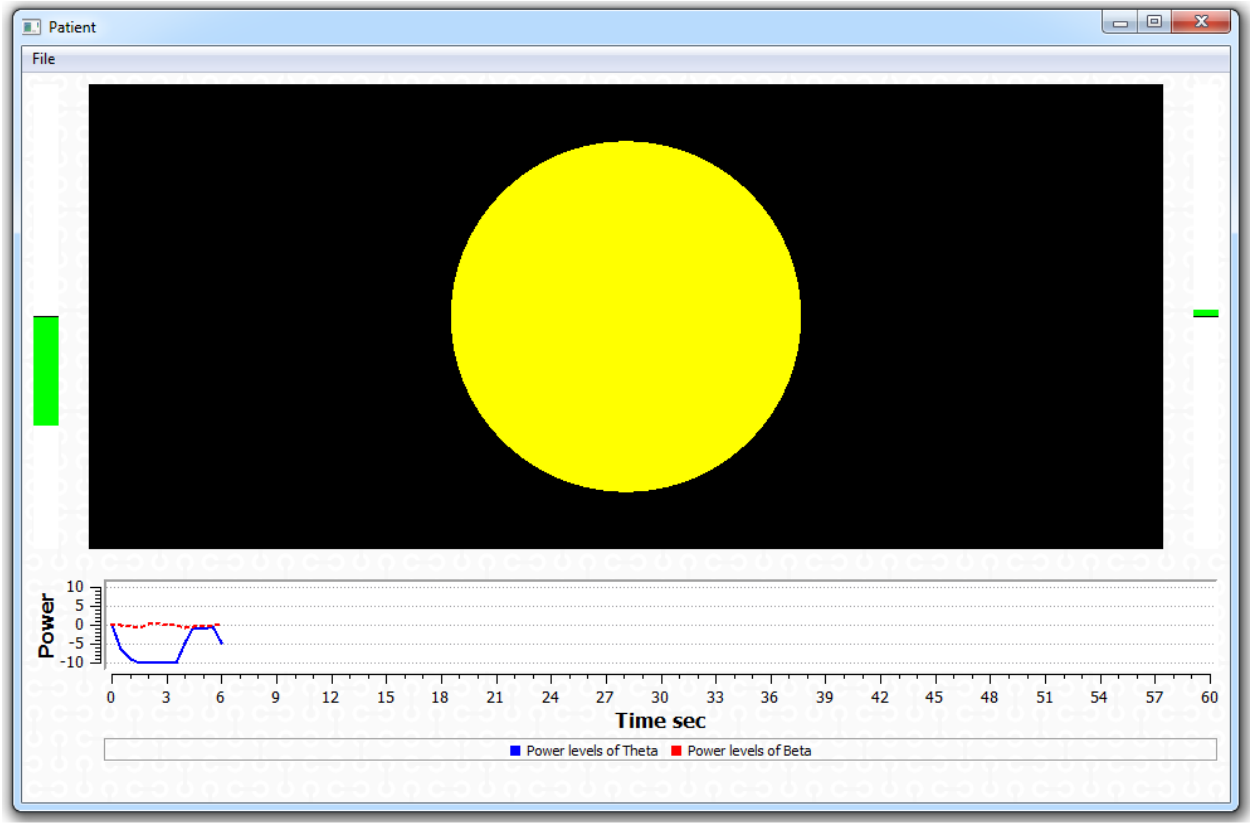
Simple Ball Game

Simple Ball Game - where patient must try to focus to move the ball up and keep it there. The higher the ball goes the better is her/his score.



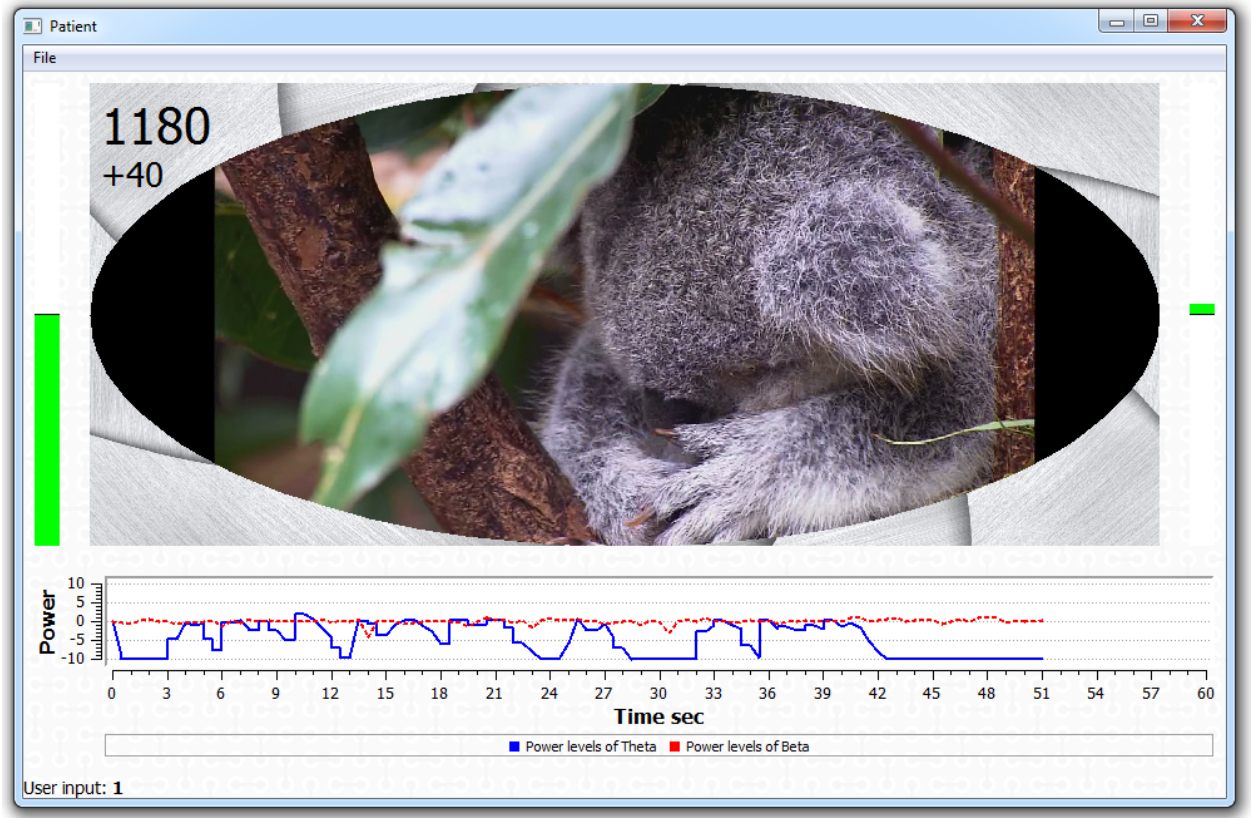
Empty Ball Game

Empty Ball Game- example of game, to show on a list. During this game circle changes color.

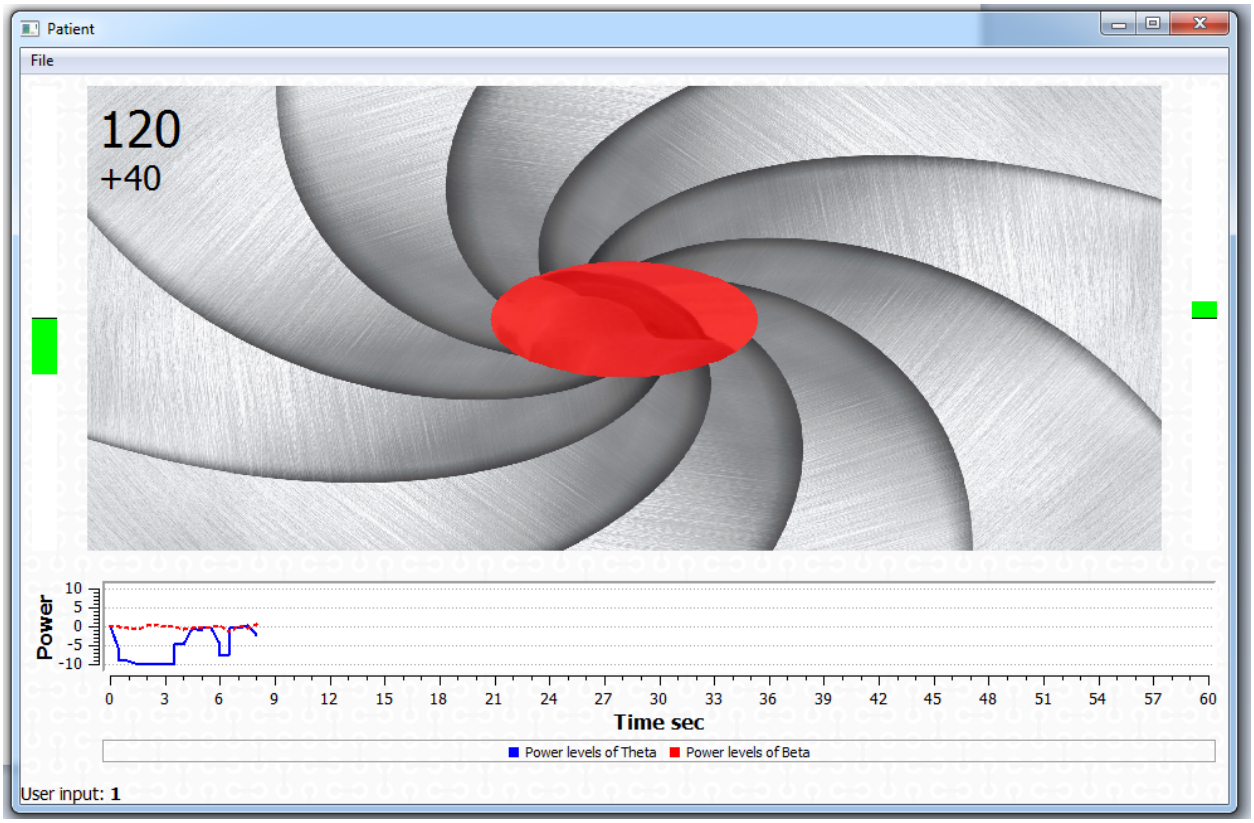


Media Game

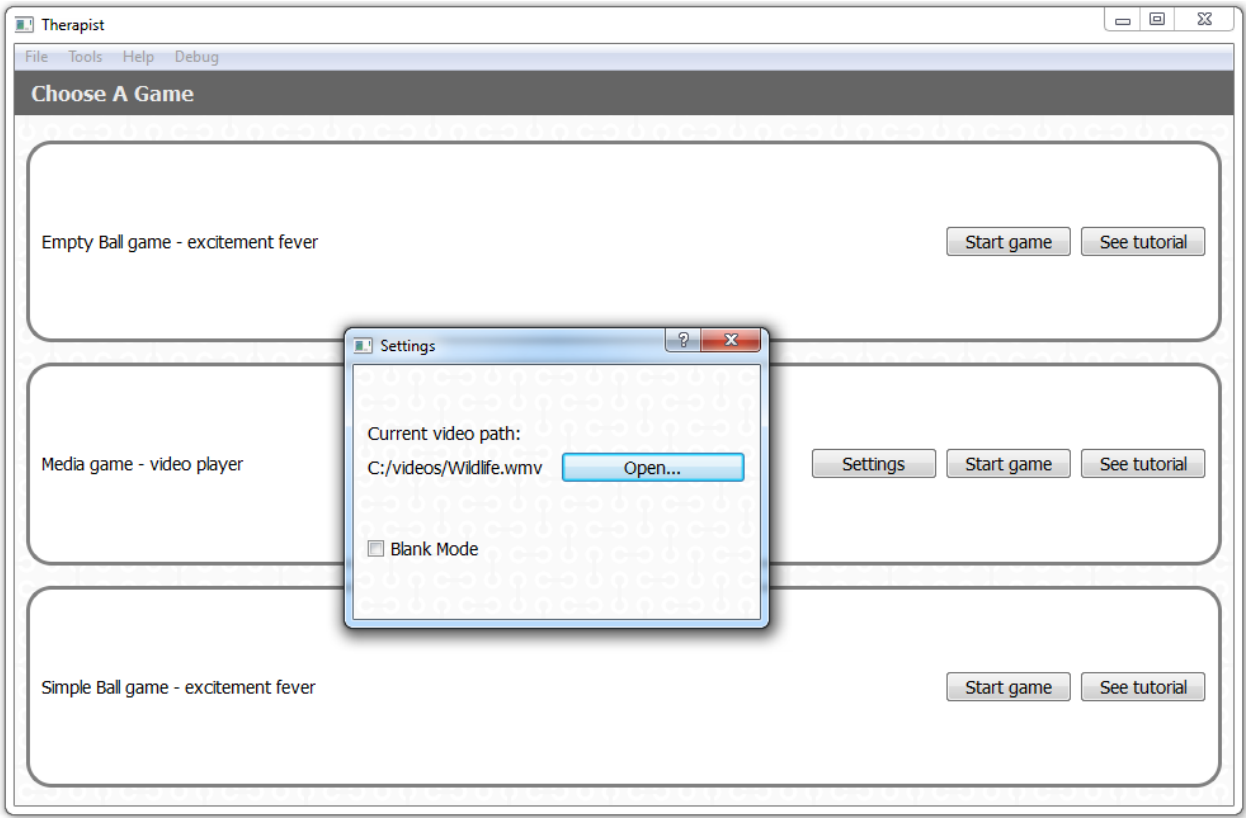
Media Game - presented video files to the patient while obfuscating them with visual noise. Level of obfuscation is calculated on the basis of the changes of EEG signal classification in time.



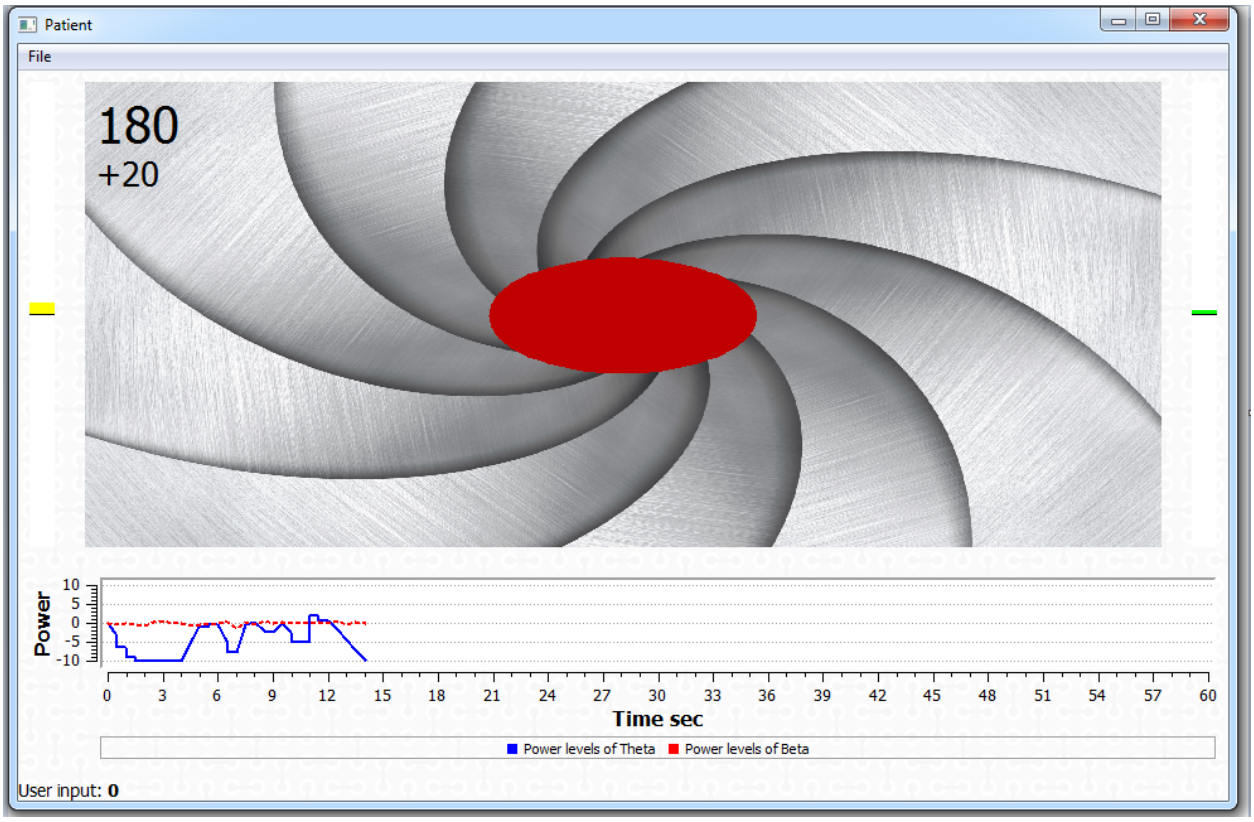
- Media Game:
  1. is showing to the user his current score
  2. is showing to the user current obfuscation level in a visual manner:
     - a as varying alpha intensity which ranges from transparent (no obfuscation at maximum high performance) to opaque (total obfuscation at maximum poor performance).
     - as a clipping view port in the shape of ellipse with neutral colored (palette of gray) screen outside it: the ellipse radius varies in steps depending on recent classifications.



3. provides settings screen for the therapist



- **Media Game Video or Image mode** Playing a video file (of common formats like ".avi") file (with audio) or image file (like ".bmp") in the center of the game screen. Video is obfuscated by alpha based obfuscation using a picture with noise or a solid color layer. Noise transparency is depend on the obfuscation level calculated by the obfuscation algorithm. There is no audio obfuscation. The video file has to be supported by Windows native player and system codecs.
- **Media Game Blank Mode** Blank screen just with all other controls, that are common in all modes (like current score for example).



All values like alpha levels, Elliptical clipping frame obfuscation can be modified in:

```
[CENT_INSTALLED_FOLDER]\plugins\MediaGamePluginData.ini
```

Example is shown bellow:

```
[General]
ConstFactor=10
TimeoutObfuscationTimeMs=2000
ObfuscationLevels=100, 85, 70, 55, 40, 25
AlphaLevels=0, 40, 77, 115, 153, 192
WeightArray=2, 2, 1, 0.1
```

2.4.2 Tutorial

All games in CENT application have tutorial which explains the rules of the game.

2.4.3 Location for the Games' Data

All game results are saved in patient's folder in:

```
My Documents\CENT\Patients\[OBFUSCATED_PATIENT_NAME]\session_[DATESTAMP]\games\
```

There is also saved session summary.

2.4.4 CENT Games API

General Information and Requirements

Exemplary CENT games are created with OpenGL and Qt. You can get Qt, SDK and free IDE at http://qt.nokia.com. There is an API for game developers, providing all needed communication from CENT to game, and back.

CENT Interface for New Games

Developers provide an interface to create new games. It is abstract class *ICentGamePlugin* in *ICentGamePlugin.h*. It is shown below:

```
#ifndef CENT_GAME_PLUGIN_H
#define CENT_GAME_PLUGIN_H

// Includes special CENT data types. Data from device.
#include "CentDataTypes.h"

/*
        This virtual class is an interface for new games in CENT application.
*/
class ICentGamePlugin : public QObject
{
        Q_OBJECT
public:
        virtual ~ICentGamePlugin() {};

public:
        // Returns name of the game. You will see this name on page
        // in CENT where you can choose a game.
        virtual QString gameName() = 0;

        /* This one is called to initialize, create and return game widget ready to start the game
        *  It is mandatory to be called before gameWidget()
        *  @return Game Widget if it is ok to start the game, otherwise NULL.
        */
        virtual QWidget* gameWidget() = 0;

        // Release Widget, is called prior to unloading the game plugin (this dll)
        virtual void releaseWidget() = 0;

        // Returns true if settings/configs can be written; returns false otherwise.
        virtual bool isConfigurable() const = 0;

public slots:
        // Called each time when the data comes from device.
        virtual void onUserInput(CentData::DigitalData data) = 0;

        // Called one time. Sets the expected input.
        virtual void onExpectedInput(const CentData::DigitalData& data) = 0;

        // Called each time when the EEG data come from the device.
```

```
41          // You can use it to display EEG plot during the game.
42          virtual void onEEG(CentData::AnalogData data) = 0;
43
44          // This method starts the game. There you have to restore to the initial
45          // state all variables and also emit gameStarted() signal.
46          virtual void onStartGame() = 0;
47
48          // Called when the game is ended from CENT ("End game" button or game time passed).
49          virtual void onEndGame() = 0;
50
51          // Called each time when the power data come from device.
52          // You can use it to display power plot during the game.
53          virtual void onPowerSignal(CentData::AnalogData data) = 0;
54
55          // This method shows the game settings.
56          virtual void onShowSettings() = 0;
57
58  signals:
59          // Copied from onExpectedSignal if not generated by game
60          void expectedInput(CentData::DigitalData data);
61
62          // Needed by the observer to start classification
63          void gameStarted();
64
65          // Decided by the game or called from onEndGame
66          void gameEnded();
67  };
68
69  Q_DECLARE_INTERFACE(ICentGamePlugin, "cent.game.plugin");
70
71  #endif // CENT_GAME_PLUGIN_H
```

You have to implement virtual methods and slots:

```
        virtual ~ICentGamePlugin() {};
        virtual QString gameName() = 0;
        virtual QWidget* gameWidget() = 0;
        virtual void releaseWidget() = 0;
        virtual bool isConfigurable() const = 0;
        virtual void onUserInput(CentData::DigitalData data) = 0;
        virtual void onExpectedInput(const CentData::DigitalData& data) = 0;
        virtual void onEEG(CentData::AnalogData data) = 0;
        virtual void onStartGame() = 0;
        virtual void onEndGame() = 0;
        virtual void onPowerSignal(CentData::AnalogData data) = 0;
        virtual void onShowSettings() = 0;
```

CENT Data Structres

There is also special class *CentDataTypes* in *CentDataTypes.h*. It is shown below. This class contain structure for data come from OpenViBE game scenario.

```
1  #ifndef CENT_DATA_TYPES_H
2  #define CENT_DATA_TYPES_H
3
4  #include <QMetaType>
5
```

```
// Structures for data from OpenViBE game scenario
namespace CENT
{
        enum ErrorCode
        {
                Success = 0,
                WrongArgumentError,
                NotSelectedCapDrivers,
                UnknownError
        };
};

namespace CentData
{
        // Max number of channels
        const int CHANNEL_MAX = 128;

        //
        enum GameControlStream
        {
                Classification = 0,
                EpochNotifier
        };

        //
        enum EEGPowerStream
        {
                POWER_THETA_LEVEL = 0,
                POWER_THETA_RANGE,
                POWER_BETA_LEVEL,
                POWER_BETA_RANGE
        };

        //
        enum CapDrivers
        {
                CapDriver4thChannels = 0,
                CapDriver8thChannels,
                UnknowCapDriver
        };

        // Cap drivers structure
        struct CapDriverData
        {
                CapDrivers type;
                QString    name;
                QString    commandToRun;
        };

        // Timestamp structure
        struct time
        {
                // Seconds
                long tv_sec;
                // Miliseconds
                long tv_usec;
        };

```

```
64          // Digital data structure
65          // Stimulations sent to CENT from OpenViBE scenario
66          struct DigitalData
67          {
68                  // Timestamp of button press/release
69                  struct time msg_time;
70                  // Which button (numbered from zero)
71                  int button;
72                  // Button state (0 = off, 1 = on)
73                  int state;
74          };
75
76          // Analog data structure
77          // Data of any stream matrix sent to CENT from OpenViBE scenario
78          struct AnalogData
79          {
80                  // Timestamp of analog data
81                  struct time msg_time;
82                  // How many channels
83                  int num_channel;
84                  // Analog channel values
85                  double channel[CHANNEL_MAX];
86          };
87  };
88
89  Q_DECLARE_METATYPE(CentData::AnalogData)
90  Q_DECLARE_METATYPE(CentData::DigitalData)
91
92  #endif // CENT_DATA_TYPES_H
```

Tutorial for CENT Games Developers

There is an archive *CENTGamesTutorial.zip* with the exemplary game and its source code. In this package you can find *README.txt* file with additional information. Besides that all sources and header files contain informative comments helpful during CENT game development.

Adding Game to CENT

Copy game's *DLL* file to `CENT\plugins` directory. You can also add game tutorial to *CENT\plugins\Tutorials\[GAMENAME]_tutorial*. Tutorial directory must contain a static html page, with *index.html* file as a minimum. CENT will automatically recognize new game after the restart.

2.5 EEG Cap Drivers Development

2.5.1 Overview

If you have EEG device other than Enobio® and want to use it with CENT, you have to develop new OpenViBE driver. The driver for the OpenViBE acquisition server is an object that interacts with hardware. Server recives information and send them to OpenViBE for processing. EEG Driver architecture overview is based on Enobio® EEG cap, as described in *Supported EEG Devices* section.

2.5.2 Data Structures

The driver works with two kind of data:

- header - not change in time, contains identifiers (e.g. information about channels).
- buffer - change in time, contains samples of channels.

The driver works at different stages of the execution, which should be implemented:

- configuration
- intialization / uninitialization
- acquisition

You can use Skeleton-generator provided by OpenViBE to generate all files needed to build the driver.

More information about creating new driver you found on http://openvibe.inria.fr/tutorial-creating-a-new-driver-for-the-acquisition-server/