Tel-Aviv University

Faculty of Engineering

School of Electrical Engineering

אוניברסיטת
תל-אביב

הפקולטה להנדסה

בי"ס להנדסת חשמל

Brain Damage Estimation Based on Auditory Spatial Perception

Project No. 15-1-1-896

Final Report

Created By -

David Iadgarov - 311335111

Supervised By -

Prof. Miriam Furst
Mr. Oded Barzely

Project executed at - N/A

➢ Overview:

This project is based on long standing research on the link between a human's spatial perception of the environment based on auditory stimulant and possible brain lesions, caused by infarct or multiple sclerosis (MS).
The goal being, achieving a cheap, quick, and simple way of detecting potential damage to the brain of a patient. Specifically, this project aimed to create a tool that brings said research to life and into the hands of doctors and professionals worldwide.

This tool was created as a simple, portable and easy to use GUI interface that brought the research to life. To achieve this the program simulates the tests, just as they were done in the research, in an accesible to the everyday man way. Data is collected and analyzed and the results are displayed for the professional to review.

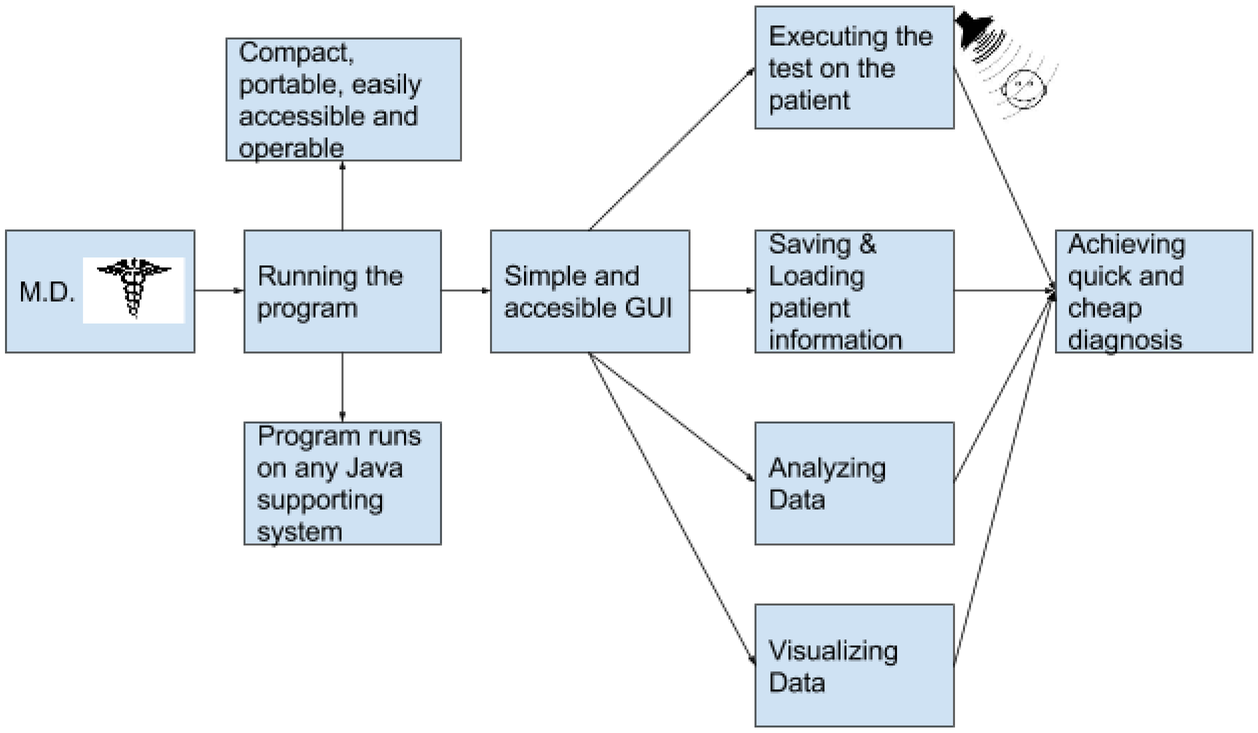

M.D.
Running the program
Compact, portable, easily accessible and operable
Program runs on any Java supporting system
Simple and accesible GUI
Executing the test on the patient
Saving & Loading patient information
Analyzing Data
Visualizing Data
Achieving quick and cheap diagnosis

Figure 1 - Block diagram illustrating project goals and highlights.

The tool was written in the java programming language. This was done in order to insure simple and easy operation on any computer system that supports Java. The vast majority of devices worldwide support and run Java, a free and readily available online software. Furthermore, all requirements and dependencies were encapsulated into the project itself, absolving the user of any need for special setup or knowledge in programming. It runs as is, anywhere.

➤ Introduction:

The goals -

This project aims to supply an easy to use, readily available means to detect potential brain damage in MS and infarct patients. To achieve this we leverage research done over many years based on the connection between auditory spatial perception and relative location of lesions in the brain.
The aim is to provide a tool that can be used by doctors with little to no prior training, no need to special equipment or expensive software.
Said tool is to be compact, portable and simple to use while allowing all necessary functionality.

The motivation-

Injuries to the brain in the form of legions are unfortunately common, be it MS patient or those suffering from the aftereffects of an infarct. The most precise way to discover these damages and their location is to use an MRI machine. Such tests are expensive and have long waiting lists that delay patient in need of help. Should we be able to supply an alternative approach, or at the least a better filter for those in need of an MRI scan, we could shorten these waiting list substantially. This tool, based on the research mentioned below, has proven itself statistically to detect these lesions in the brains of patients. Serving as a viable first option, ready, cheap and available for use at any clinic.

How we went about offering a solution-

Creating a simple to use tool that can simulate the tests from the aforementioned research would allow any doctor to estimate the location of possible brain lesions for any patient. Therefore the goal was to create a program that supports the execution of what the papers discussed.
A program such as this can then be shared easily, used anywhere with little prerequisite to allow for widespread cheap detection of potential damage to the brain.
To create this program classic Java UI design was used. Primarily the Swing library.

Comparing to existing such tools-

As far as this student is aware no such tools are available. The research is well known and respected and is now finally being put to possible use.

➤ Theoretical background:

The sources for this project are twofold. First the papers describing the original research that lead to such a tool being possible and second the Java libraries and documentation which served to bring it to life.

Research into auditory spatial perception-
Two papers were used for the development of this program:

“Sound lateralization and interaural discrimination.C E¡ects of brainstem infarcts and multiple sclerosis lesions” [1]

Lateralization and binaural discrimination of patients with pontine lesions [2]

These publications describe the method making this program possible. Having two ears, humans are able to detect the direction from which sound comes. It was noted that those with damage to certain areas of the brain had trouble replicating this normal reaction. Based on this observation an experiment was devised. The subject is exposed to auditory stimulants, sounds or several types and was told to reply as to where the sound was coming from. This correlation between the simulated sound, meant to simulate noise from a certain direction, and the subjects inputs is then compared to that of the control group. The control group consists of data collecting in the same manner from healthy individuals.
Using statistic analysis certain patterns were discovered and later used to estimate the location of possible brain trauma.
Ultimately a quick and simple method for matching pattern in auditory spatial perception to possible lesion location was developed.

In this project we took the data and method from these publication and brought them together into a usable tool. This achieve this we created a simple to use GUI. Using classic UI elements and barbones Java libraries. The backbone of this code was the Java Swing GUI library [3].
As mentioned above, the goal was to make this compact and simple as can be, workable in every environment.

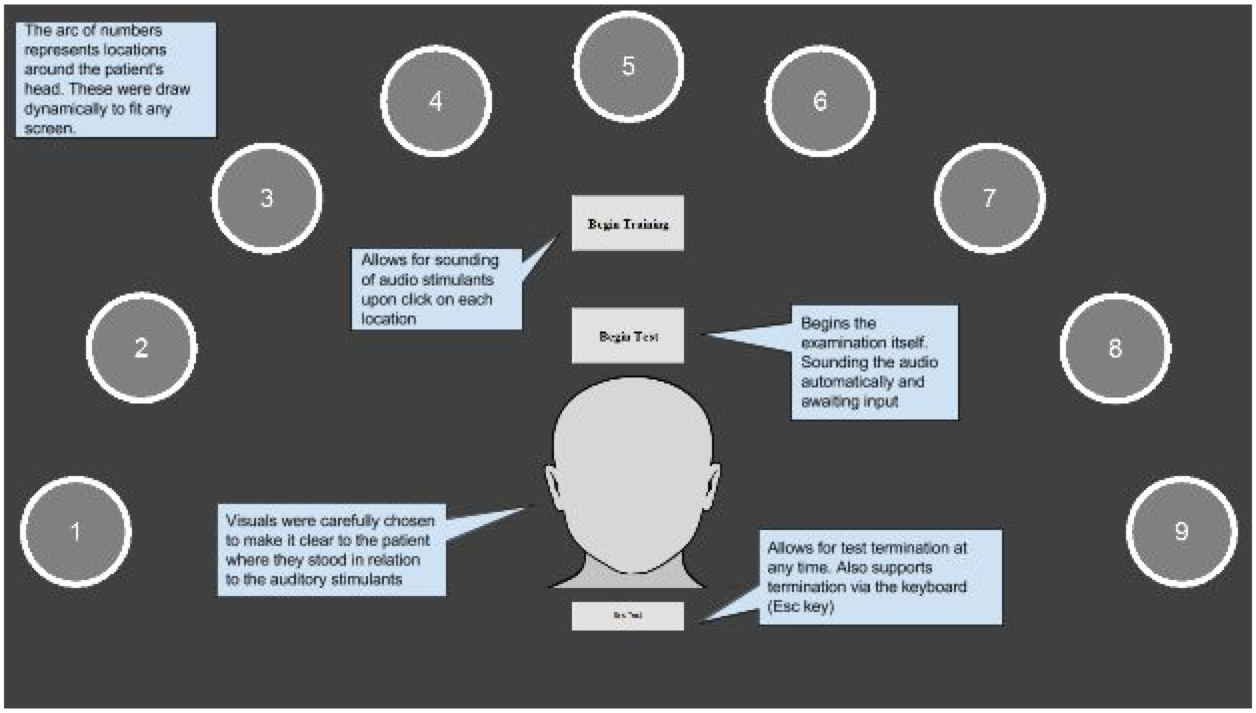
➤ The Implementation and Software:

In this section we will discuss the program, how it was written and why it works.
The program supports some key features:

1. Execution of the experiment
2. Viewing of the data
3. Saving and loading patient information
4. Customizable settings, saved from run to run.
5. Loading of custom audio stimulant files.
6. Quick navigation between the various options.

The Experiment -

Being the crux of the matter the test execution was given high priority. Written to support inputs of various types, from clicks to keyboard. A minimalistic design was chosen to remove unnecessary visual stimulants.
The screen was made to construct itself dynamically, fitting any sized screen on any machine. Additionally it is made to run in full-screen mode to further remove unwanted visual stimulants.



The arc of numbers represents locations around the patient's head. These were draw dynamically to fit any screen.

Allows for sounding of audio stimulants upon click on each location

Begins the examination itself. Sounding the audio automatically and awaiting input

Visuals were carefully chosen to make it clear to the patient where they stood in relation to the auditory stimulants

Allows for test termination at any time. Also supports termination via the keyboard (Esc key)

Figure 2.1 - Examination screen

Both the test itself and the training, the opportunity to listen to the auditory stimulant as preparation, are supported and mutually exclusive. Specific action listeners in the Java API were used to achieve this, one cannot run a test while training and vica versa.
The execution of the test begins once the user clicks the "Begin Test" button, invoking a listener nested in the code that begins the transaction. Sound stimulants are given and user input is awaited. This input can come in the form of clicks on the dynamically located buttons using ClickListener or from the keyboard by utilizing KeyListeners attuned only to the relevant keys (the numbers, 1-9).
As the the test runs the input is documented in parallel as to not interrupt the process.

As mentioned above the program offers the option to run a 'training' session before the test begins. This allows the user to discover the auditory stimulants and become accustomed to them before the start of the examination itself. An important stage seeing as the brain perceives stimulants by comparing to some benchmark it has experienced in the past.
To achieve this the user must be able to play each stimulant for each location in the arc. Seeing as our stimulants are twofold, a set of time delayed signals and a set of level modified ones, a choice must be clearly presented to the user. This is achieved by using two, initially hidden, buttons that come into play once a training sequence begins. These buttons indicate which of the stimulants has been chosen and are inactive and invisible so long as the training is not undergoing. See Figure 2.2.

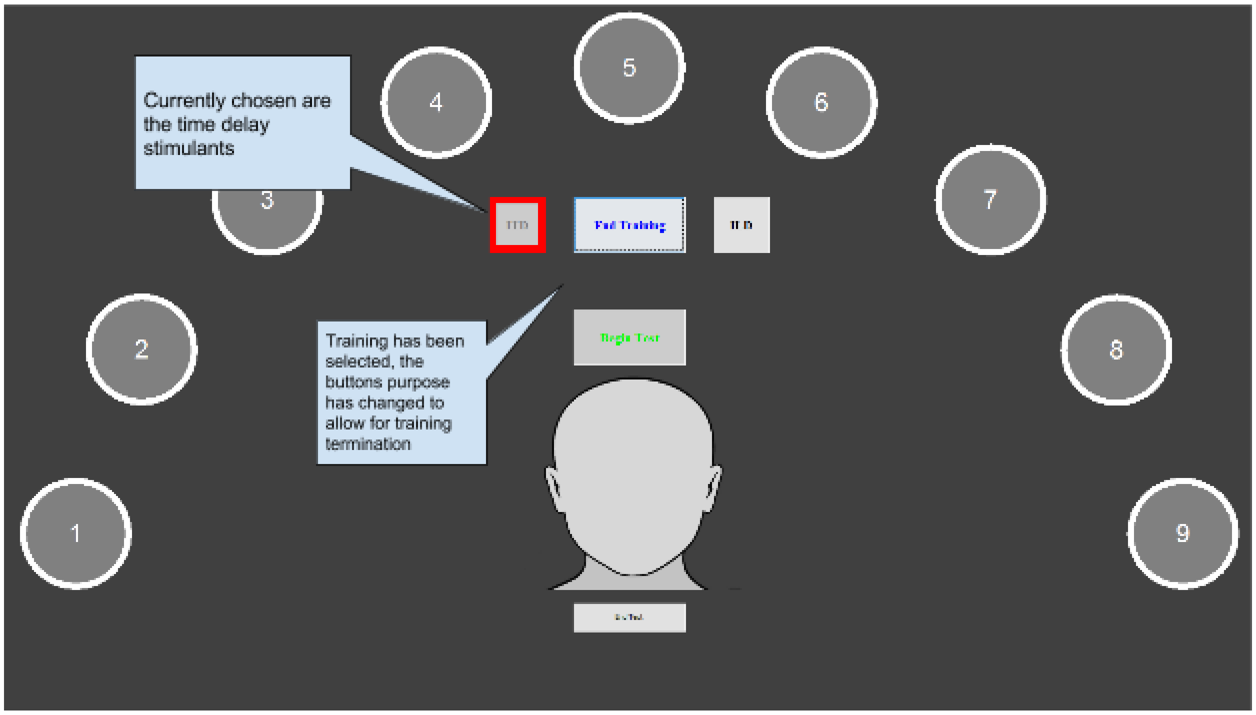

Currently chosen are the time delay stimulants
1
2
3
4
5
6
7
8
9
TTD
End Training
ILD
Begin Test
Training has been selected, the buttons purpose has changed to allow for training termination

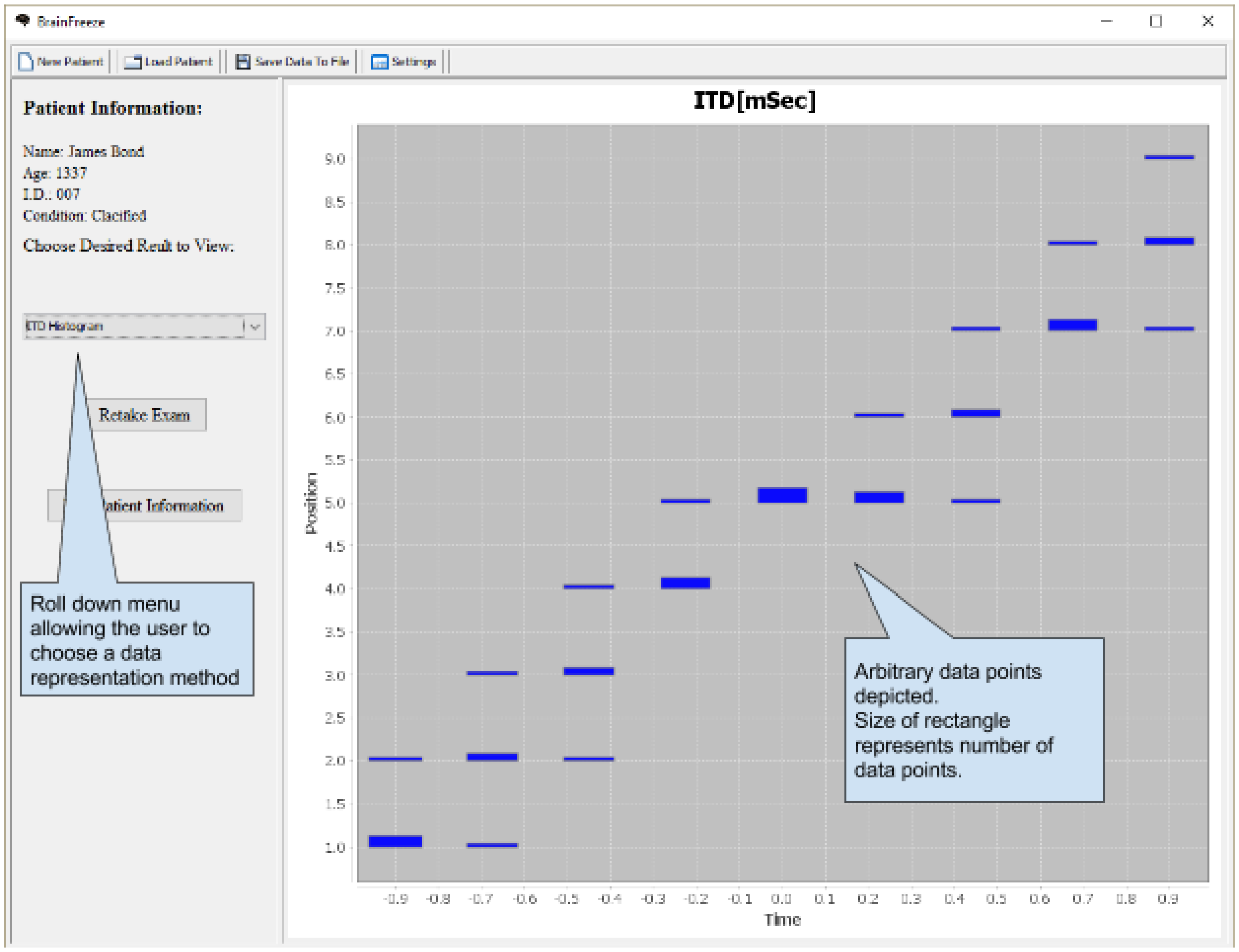
Figure 2.2 - Training

The Data -

To allow for proper collection of data points several fronts had to be covered. The raw information itself was fed into a custom made class, the ‘Measurements’ class. Any object of said class contains the information relevant to it and various methods for analyzing the data. Each patient enrolled into the program is given a corresponding patient object in the code, as mentioned before. Every such object contains within itself a Measurement object. Thus, every collection of measurements belongs to its relevant patient.
Information is driven into this object as the user generates it. Inside the test window, as mentioned above, lie many ActionListener that record any choices made by the user into the relevant Measurements object.

Once collected the data becomes available to the user, to be viewed in several manners. First, and most important are the graphs. These are histograms that match stimulants belonging to certain locations to the chosen locations by the user. A healthy humans input should result in an even and linear distribution.
Two variants of the histogram exist, one per stimulant type (time delayed signal or level altered signal). See figure 3.
To generate the graph another external library had to be used. This additional source offered a basic structure for graphing in Java. In order to accommodate the specific type of histogram required here some original code had to be layered on top of said library. By overwriting and adding to the library and graph entity such as the one we want became possible to generate.
Additionally there exist two more options for viewing the results. A pane dedicated to the “dry calculations”, including regression lines, probability calculations and the like and a pane for final diagnosis.

Diagnosis is achieved by placing every patient into one of three sets. Healthy, center oriented, or side oriented.
A person with no damage to the brain or hearing will in the vast majority of cases fall into the healthy set. The rest are distinguished between the other two.
Center oriented people tend to choose locations near the center of the head as the area from which the sound stimulant came from, often for sounds that do not in fact come from the center.
By contrast, side oriented results point to people who find it difficult to discern when a sound comes from a location close to their head and instead tend to prefer marking stimulants as polarized to one of the sides (right or left).
Each of these can be spotted quickly and easily by the operator by taking a look at the histograms. An evenly distributed line points to a healthy individual, a dense collection at the extremities points to side orientation while a large collection of data in the center points to a center oriented patient.
Once it is apparent which of the three sets a patient belongs to assumptions can be made as to the location of the potential brain damage.


BrainFreeze
New Patient
Load Patient
Save Data To File
Settings
Patient Information:
Choose Desired Result to View:
ITD Histogram
Retake Exam
Patient Information
Roll down menu allowing the user to choose a data representation method
ITD[mSec]
Position
Time
Arbitrary data points depicted. Size of rectangle represents number of data points.

Name: James Bond
Age: 1337
I.D.: 007
Condition: Clacified

Figure 3 - Showing the histogram for time delayed stimulant based on a patient's test results.

All the calculations and results are of course relative. In order to receive results that work as intended they must be compared to a relevant control group, a set of healthy people. The tool comes preset with the numbers relevant to the default stimulants. Should the user want to use other stimulants the option to tweak these numbers exists in the settings screen, as will be detailed below.

Creating, Loading and Saving Patient Data -

A new patient can be easily enrolled into the program by choosing the relevant option available on the toolbar (more on that below).
It is imperative to be able to collect the generated results and keep them stored for further use. This was implemented by the creation of a custom file type, the ".patient" file, which contains all the information relevant for a patient. This includes the name, ID, medical condition, data collected through the examination, notes added by the supervising professional etc. Further, the program supports saving in this format and

loading back from it. This allows for easy sharing of data between different medical specialists, the files is saved and can be reloaded on any machine using our tool.
A large goal was simplicity and flexibility, hence the program makes sure to not allow illegal inputs or outputs. Should the operator, for instance, forget the ".patient" extension of the file it is added automatically, similarly the loading screen does not display files of formats it does not support. Should the user attempt to load corrupted patient file an error message is displayed as the file does not pass assertion.
Here too the logic executes immediately upon user invocation with eh use of action listeners running in parallel with the program.

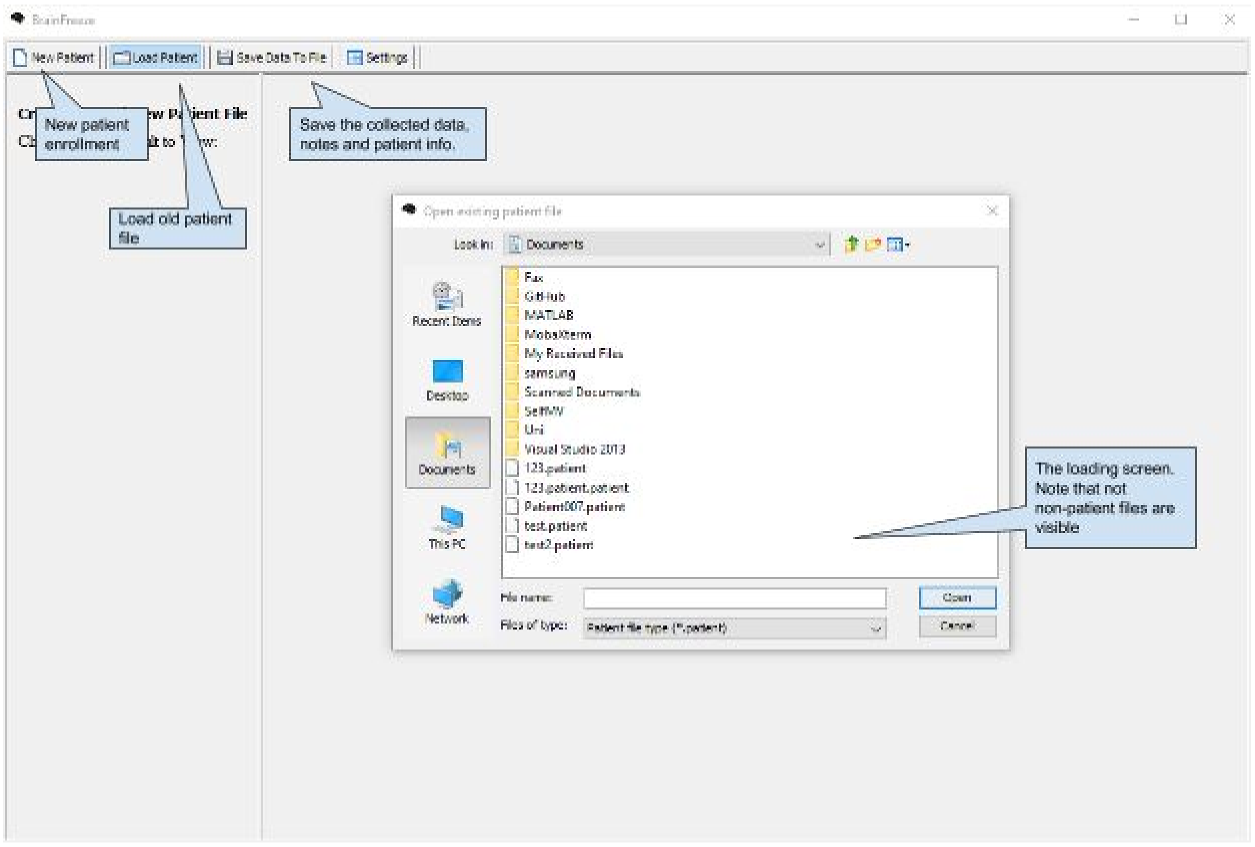
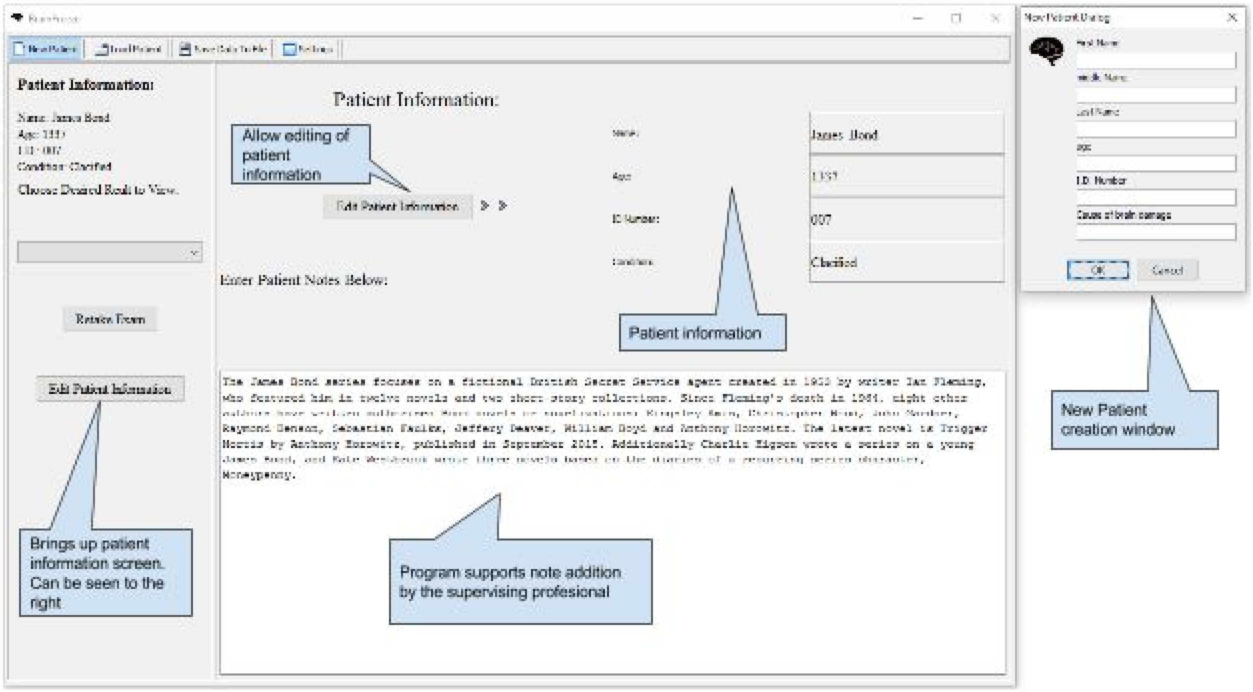


Figure 4 - Creating, Loading and Saving Data



Patient Information:

Name: James Bond
Age: 1337
I.D.: 007
Condition: Classified
Choose Desired Result to View:

Patient Information:

Name:	James Bond
Age:	1337
ID Number:	007
Condition:	Classified

Enter Patient Notes Below:

```
The James Bond series focuses on a fictional British Secret Service agent created in 1953 by writer Ian Fleming,
who featured him in twelve novels and two short story collections. Since Fleming's death in 1964, eight other
authors have written authorised Bond novels or novelisations: Kingsley Amis, Christopher Wood, John Gardner,
Raymond Benson, Sebastian Faulks, Jeffery Deaver, William Boyd and Anthony Horowitz. The latest novel is Trigger
Mortis by Anthony Horowitz, published in September 2015. Additionally Charlie Higson wrote a series on a young
James Bond, and Kate Westbrook wrote three novels based on the diaries of a recurring series character,
Moneypenny.
```

Figure 5 - Patient Data and Creation screen

Program Settings -

The program supports various options that customize the test taken by the patient. Amongst these are the delay time between audio stimulants, the audio stimulants themselves, the nature of the transaction between stimulants. Number of data points to collect per stimulant and advanced editing of statistical data on which the data analysis depends.

Transition between sounds during the testing is possible both on a set delay, which can be altered by the user, and upon the user's prompt (click).
A balance was needed between the number of user inputs per audio stimulant (translating into the number of times said stimulant was invoked) and the length of the examination. The papers suggest 4 data points per stimulant. The settings screen offers the option of changing the number of data points should the user decide it is needed.
Next the user is offered the option of using custom audio stimulants. To achieve this ".wav" files are loaded from a directory specified by the user. The directory search happens in an interactive loading screen for the sake of user friendliness.
Furthermore an info button was added, prompting the user when clicked and specifying the acceptable format for the audio input files.
Lastly the option for advanced data analysis tinkering is given. The results depend on the results of the control group, should new control data be available it can also be used. These settings are protected by a 'tick box', disabling the ability of the user to edit them unless specifically requesting the access.

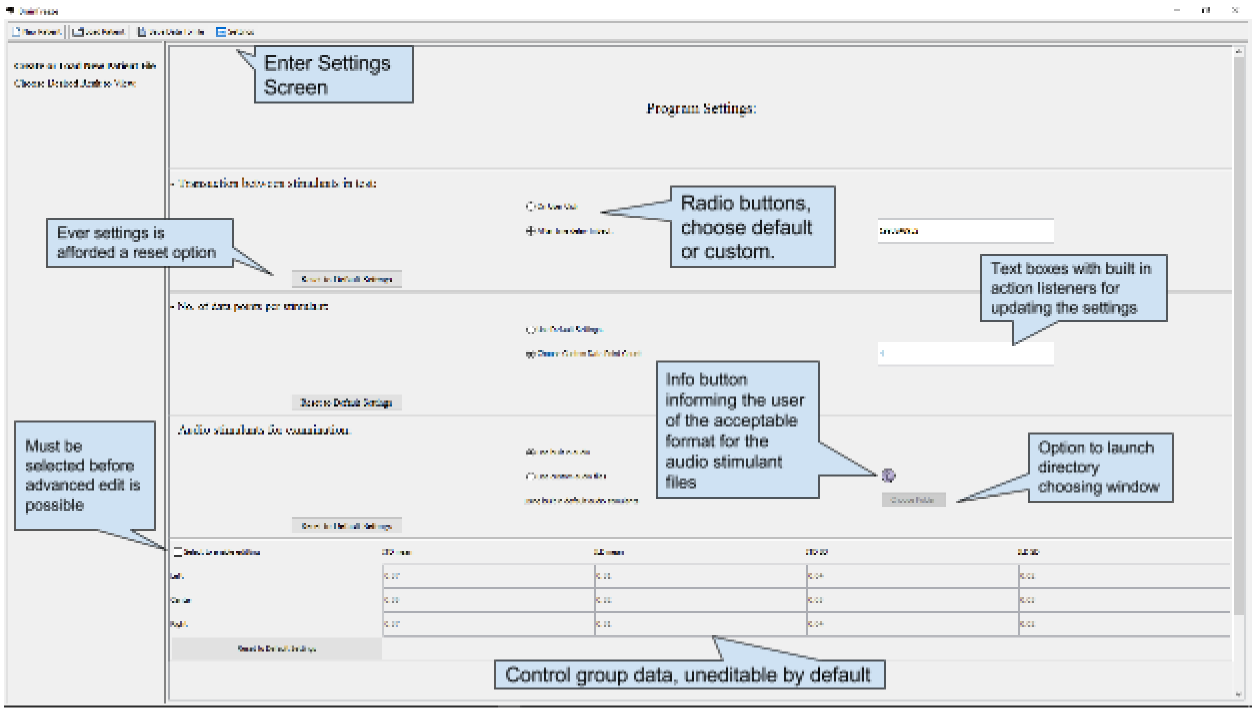

Enter Settings Screen
Program Settings:
Radio buttons, choose default or custom.
Ever settings is afforded a reset option
Text boxes with built in action listeners for updating the settings
Info button informing the user of the acceptable format for the audio stimulant files
Option to launch directory choosing window
Must be selected before advanced edit is possible
Control group data, uneditable by default

Figure 6 - Settings screen

As before all actions are listened for via action listeners from the Java Swing API. Settings that may be dynamically updated are given a textbox for setting custom values in.
All settings are saved as soon as they are set. A special settings file format was created the “.settings” file. Should a settings file not be available the program reverts to the defaults.
Every setting was afforded a reset button, allowing the user to factory reset said setting.

Creating, Using and Loading the Audio Stimulants -

Naturally the sounds presented to the patient are an imperative part of the process. As such they were carefully created and played.
The program supports both hard coded default sounds, for the sake of encapsulation and runnability of the program anywhere while still allowing flexibility and choice to the operator.
The default sounds were created using MATLAB with the help of the project supervisors and made to fit the specifications of the research on which this tool leans. A “.wav” file format was chosen, being an audio file it encodes in itself all the necessary data, from the samples to the sampling rate.
The default sounds that were chosen are short clicks. Half with a channel (right or left) delayed in time to simulate different spatial perception of the sound and half with level differences between the two channels for the same reason.
When a sound appears weaker to our left we naturally assume it came from the right. Similarly a sound that we hear first on the right and then a split second later on the left must have come from the right (one ear is closer to the source than the other). This is the reasoning behind the manner of sound generation described here.

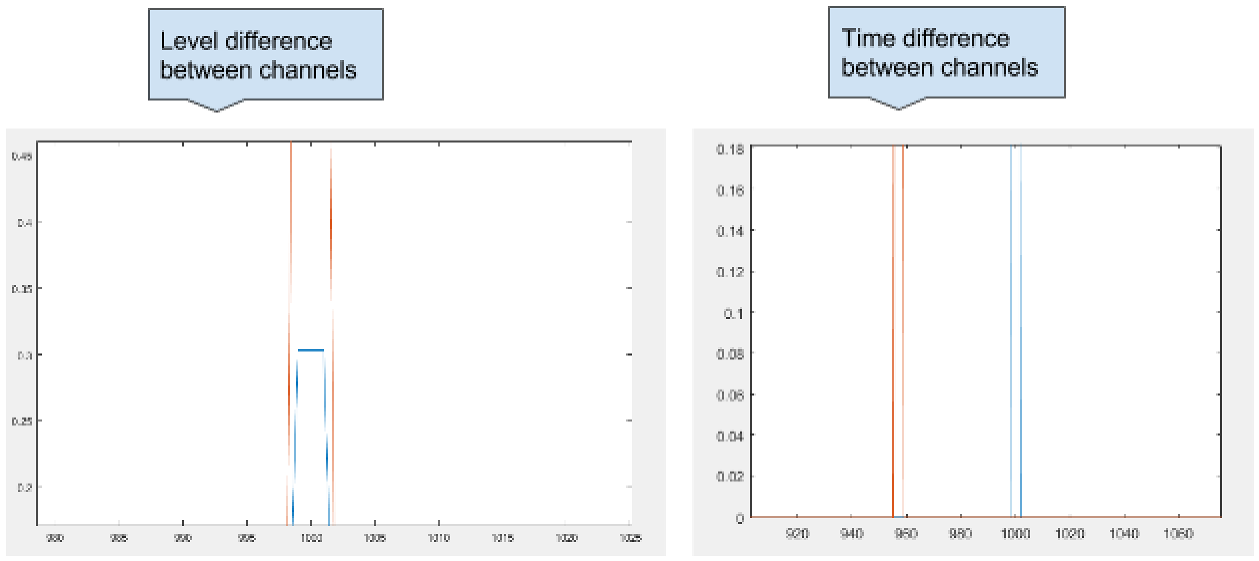
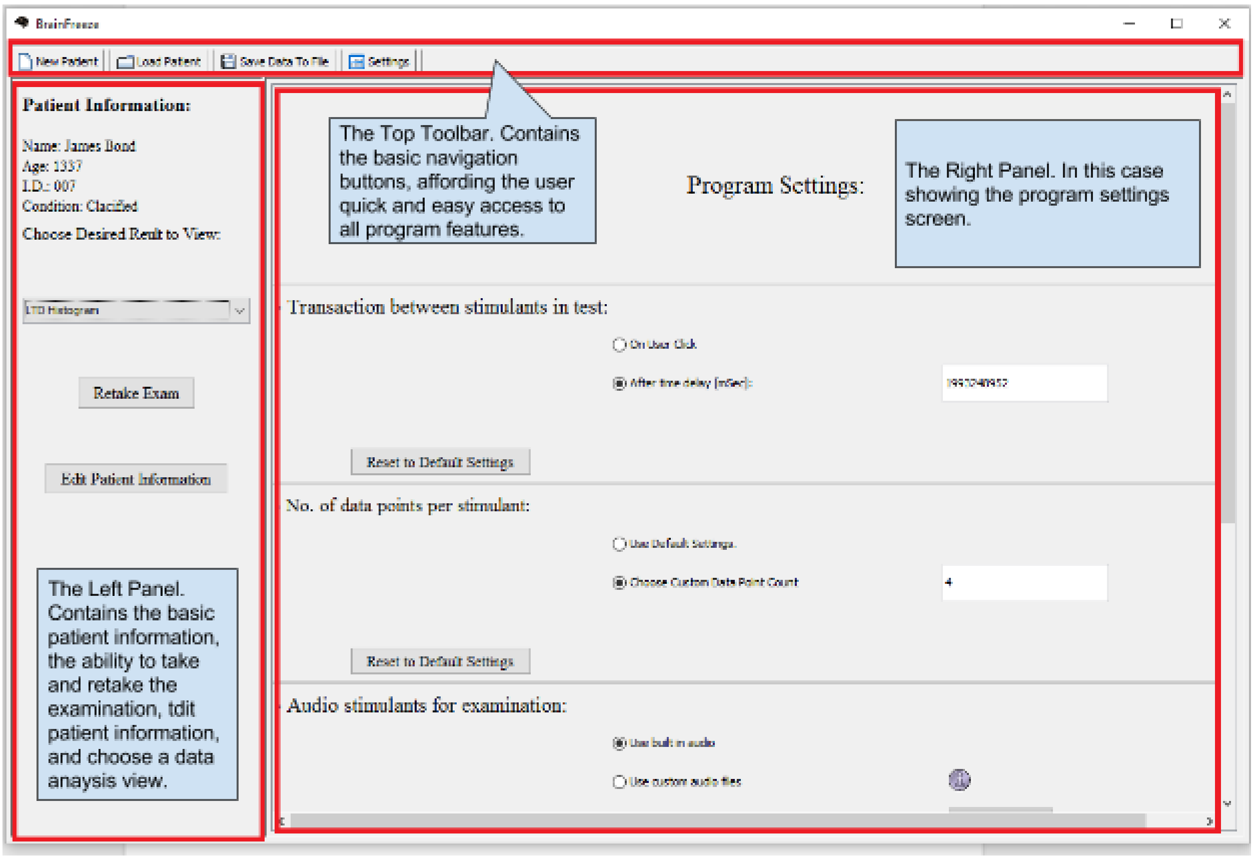


Figure 7 - Default Sounds - Short Clicks - Level Difference (ILD) [left] and Time Difference (ITD) [right]

To play these sounds back during the examination or training the Java sound library was used. The code connects to the system audio stream and using the Clip class pushes the data to the audio output of the machine whenever prompted.
In order to support both file-based audio and the hard coded audio (which exists as a resource inside the source code) the polymorphism aspect of object oriented programing was leveraged. The program loads a general “audio source” which is implemented in two ways, once as a file resource and once as a project resource. Thus both option work just the same when the time comes to play the sound.

Navigation and Design of the Program -

The program GUI consists of three main segments. The top toolbar which serves for easy navigation. The left panel which depicts the patient and action that may be performed on them and the right panel, a dynamically interchangeable “main screen” of the program. In the right panel live the data, settings screen, patient information, etc.



Patient Information:

Name: James Bond
Age: 1337
I.D.: 007
Condition: Clacified

Choose Desired Reult to View:

The Top Toolbar. Contains the basic navigation buttons, affording the user quick and easy access to all program features.

Program Settings:

The Right Panel. In this case showing the program settings screen.

Transaction between stimulants in test:

- ( ) On User Click
- (●) After time delay [mSec]: 1993240952

No. of data points per stimulant:

- ( ) Use Default Settings.
- (●) Choose Custom Data Point Count: 4

Audio stimulants for examination:

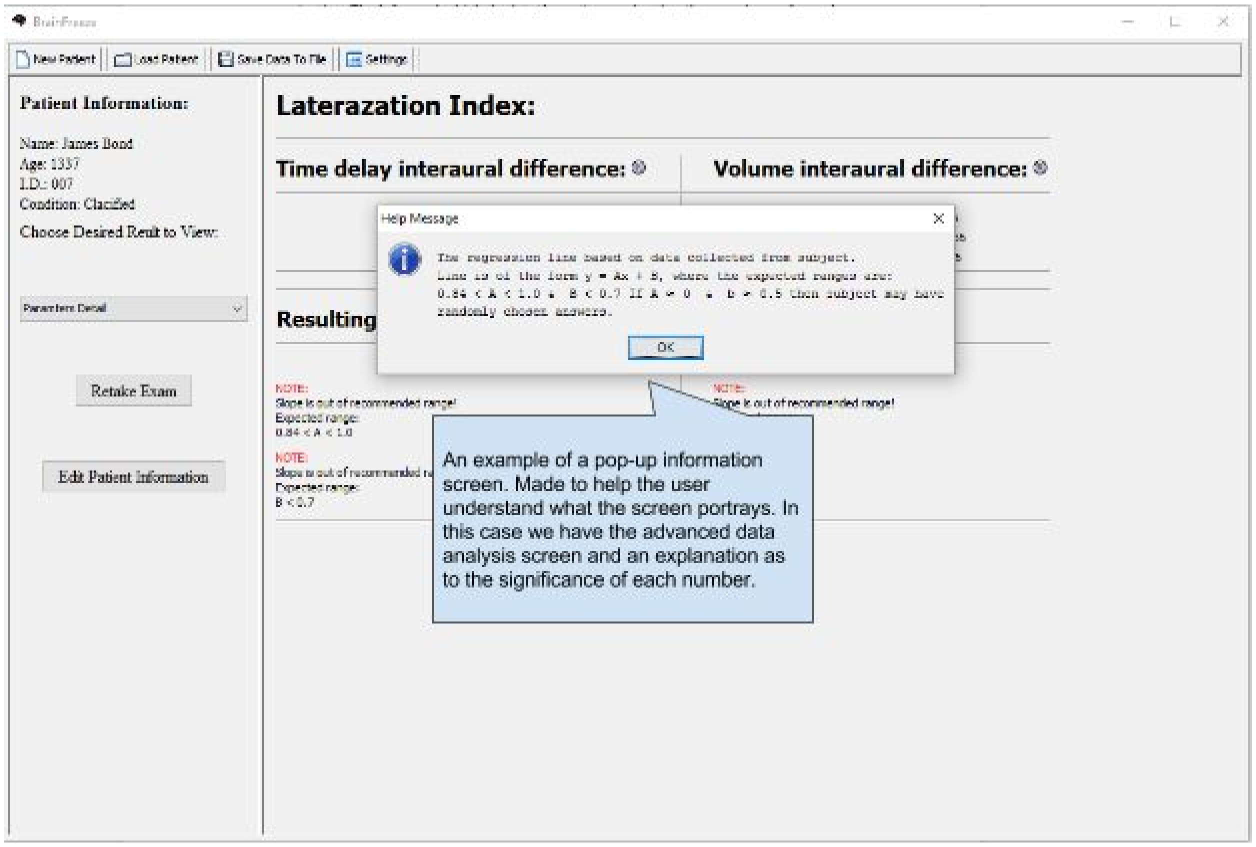
- (●) Use built in audio
- ( ) Use custom audio files

The Left Panel. Contains the basic patient information, the ability to take and retake the examination, tdit patient information, and choose a data anaysis view.

Figure 8 - General form of the UI

This, along with every other screen of the program was created with the use of clever GUI layout managers. Throughout the program several different layouts were utilized, from grids to amorphic relative layouts and custom made graphic drawing methods (for instance the test screen).
The general design was created while keeping in mind user friendliness. Illegal actions will not crash the tool, instead an error message will be displayed. The UI allows to changes in size (for instance the left and right panel are resizable) to accommodate the user's preference.

In addition to the main program interface several pop-up screen were added. For instance the saving screen, the loading for both patient and audio files, and various error messages or information prompts. It is worth noting that these were made in a way that prevents the user from interacting with the main program process until they are dismissed. Avoiding bugs and unnecessary complications.



Patient Information:

Name: James Bond
Age: 1337
I.D.: 007
Condition: Clacified

Choose Desired Reult to View:

Parameters Detail

Laterazation Index:

Time delay interaural difference:

Volume interaural difference:

Resulting

Help Message

```
The regression line based on data collected from subject.
Line is of the form y = Ax + B, where the expected ranges are:
0.84 < A < 1.0 & B < 0.7 If A < 0 & b > 0.5 then subject may have
randomly chosen answers.
```

NOTE:
Slope is out of recommended range!
Expected range:
0.84 < A < 1.0

NOTE:
Slope is out of recommended r
Expected range:
B < 0.7

NOTE:
Slope is out of recommended range!

An example of a pop-up information screen. Made to help the user understand what the screen portrays. In this case we have the advanced data analysis screen and an explanation as to the significance of each number.

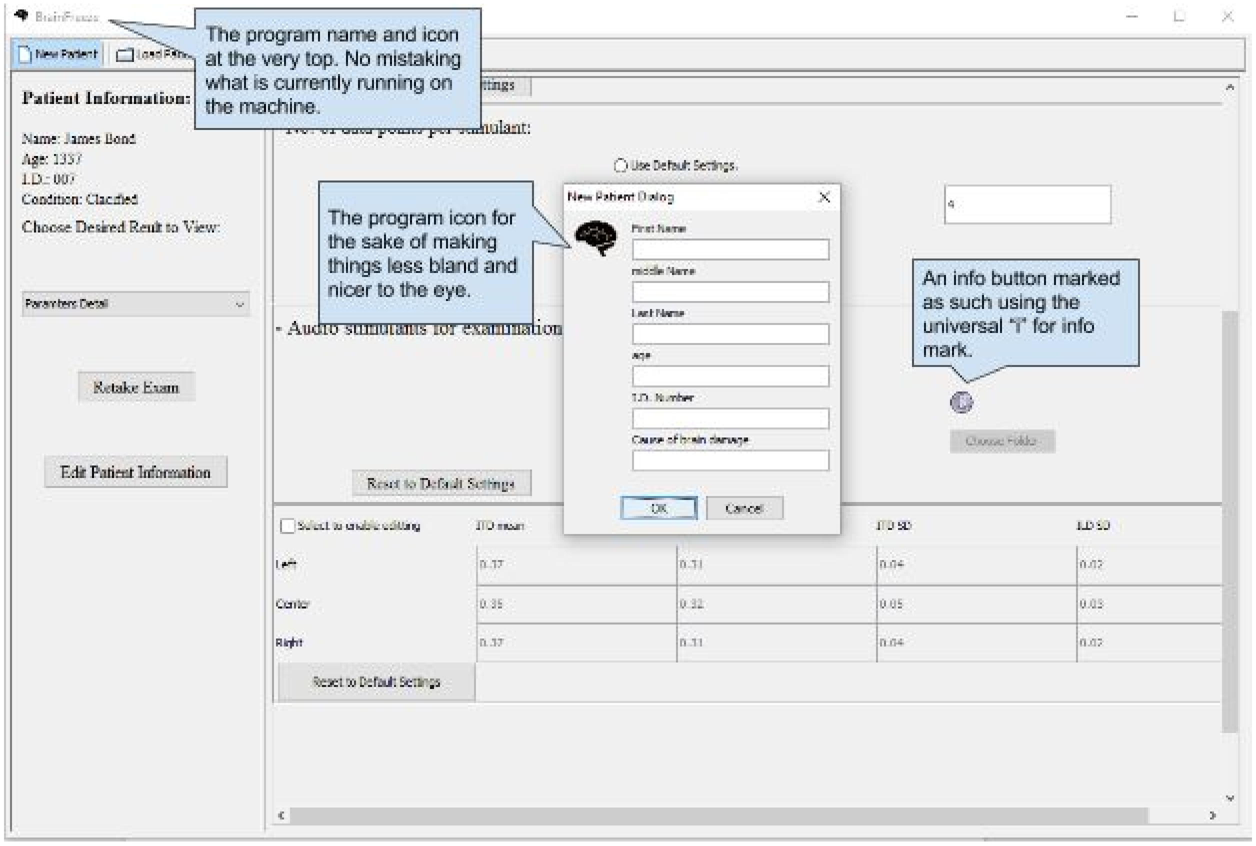
Figure 9 - An instance of a pop-up message, designed to help the user operate the tool

The program was also given some minimalistic beautifications. Aspects like icons, and intuitive looking buttons. These serve to make the interface more friendly to the user and simple to understand. Some examples can be seen in Figure 11 below.

Additionally an icon for the program launch file was desired. As this was written in Java the run file is a “.jar” file, which do not support custom icons. To remedy this advanced tools were used to wrap the jar inside of an executable file (“.exe”) and an icon was then superimposed onto it.

Said icon was custom made for this program using GIMP [4].

Figure 10 - The program icon:


BrainFreeze

New Patient | Load Patient

The program name and icon at the very top. No mistaking what is currently running on the machine.

Patient Information:

Name: James Bond
Age: 1337
I.D.: 007
Condition: Clacfied

Choose Desired Reult to View:

Parameters Detail

Retake Exam

Edit Patient Information

No. of data points per stimulant:

Use Default Settings.

- Audio stimulants for examination

The program icon for the sake of making things less bland and nicer to the eye.

New Patient Dialog

First Name
middle Name
Last Name
age
I.D. Number
Cause of brain damage

OK | Cancel

An info button marked as such using the universal "i" for info mark.

Choose Folder

Reset to Default Settings

Select to enable editing	ITD mean		ITD SD	ILD SD
Left	0.37	0.31	0.04	0.02
Center	0.35	0.32	0.05	0.03
Right	0.37	0.31	0.04	0.02

Reset to Default Settings

Figure 11 - User friendly UI aspects

➤ Summary, Conclusions and Suggestions for the Future:

What we have here is the culmination of research, code and good intentions. By taking all three a tool that could potentially save doctors and patients a lot of money and time. While MRI technology is impressive and vital to today's healthcare system it is also expensive and often blocked behind an extensive waiting list. Having an alternative way of testing for damage and estimate its location will help circumvent the need for the MRI.

The tool, while simple conceptually, brings this useful ability into the hands of the masses. As mentioned before, the whole project was written to be portable and simple to operate in any conditions. A single file, no installation required and fully available default settings allowing the user to run it without any preparation.

The potential for this tool is large. Upon this initial implementation many more features may be added. For instance some integration with the medical aspect, perhaps a rendering of the brain that highlights the potentially damaged location based on the test results.
Should the research be further developed changes can also be easily made to accommodate it, thus keeping the tool up to date and reliable.

We learn through this project of the power object oriented programming holds when it comes to GUI coding. The entire visual program can be broken down to a hierarchy tree of windows, panes, buttons, etc. OO programming allows for a very natural integration of these various components within one another.

➤ References:

[1] Sound lateralization and interaural discrimination.C E¡ects of brainstem infarcts and multiple sclerosis lesions
Miriam Furst , Vered Aharonson , Robert A. Levine , Barbara C. Fullerton , Rina Tadmor , Hillel Pratt , Andrey Polyakov , Amos D. Korczyn

[2] Lateralization and binaural discrimination of patients with pontine lesions
Vered Aharonson and Miriam Furst, Robert A. Levine Eaton Peabody, Michael Chaigrech, Amos D. Korczyn

[3] Package javax.swing,
https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

[4] GIMP - image manipulation program - https://www.gimp.org/about/