T e c h n i c a l   U n i v e r s i t y   o f   C r e t e

S c h o o l   o f

E l e c t r o n i c   &

C o m p u t e r

E n g i n e e r i n g

Diploma Thesis

"Real-Time Visualization of Neuroscience Model Simulation"

by Mytakis Dimitris

Thesis Committee

Professor Apostolos Dollas

Professor Dionisios Pnevmatikatos

Associate Professor Katerina Mania

Chania, Crete

Spring 2016

Dedicated to
My Family

Thesis Abstract

The large amount of information accumulated over the years and the research on the human - or animal – brain, requires simulations and ways to represent the data. These are necessary both for scientists and non-specialized users, in order to understand better the functions and the way that the brain manages any information it acquires.

In the contexts of this diploma thesis, an application has been designed in order to visualize the information obtained after a neural network simulation. The implementation of the Neural Network Simulation is detailed in the diploma thesis titled "Neural Network Simulation Speedup based on Reconfigurable Logic". The application also enables the input of simulation data, thus acting - in combination with the software system for simulation - as a comprehensive study tool and neural network representation.

Due to the fact that biologists who work in the neurobiology field are using standard forms for displaying exit data of a simulation, in this thesis a different approach is being tested. This approach constitutes an effort on the imaging of a non-photorealistic scene, based on scientific discoveries and microscopic photos.

In opposition with the current forms of displaying neuron data - such as tables, networks with nodes, graphs - a three-Dimensional space is visualized, by using the game engine Unity. Three-dimensional biological models coexist with some of the forms of displaying, as mentioned above.

Two forms of visualization are implemented. The first is based on the idea of raster-plot, but it is a little bit more complicated and more analytic, as it contains the value of cells and dendrites action potential. The second form of visualization was based on the idea of the creation of a three-dimensional world, with three-dimensional neuron cells models in it, and graphs with each neuron and dendrite action potential.

Thesis Acknowledgements

First of all, I would like to thank my thesis supervisor professor Apostolos Dollas for the confidence that showed in me and his faith in my abilities, associate professor Katerina Mania for her advice and professor Dionisios Pnevmatikatos for his participation in the committee.

I would also like to thank the members of the Computational Biology Lab of IMBB / FORTH for their advice on the implementation of the application (especially Athanasia Papoutsi for the time spent in evaluating the application), professor Chris Belbas for the time he spent in evaluating the application and Postdoctoral Researcher George Alex Koulieris for his proposals at the beginning of the implementation.

I would like to give special thanks to MSc Student Manolis Kousanakis for his precious help and for the time he spent during the entire thesis preparation and Vaitsa Dinopoulou for her advice in the aesthetic part of the thesis and her help in the writing of the thesis.

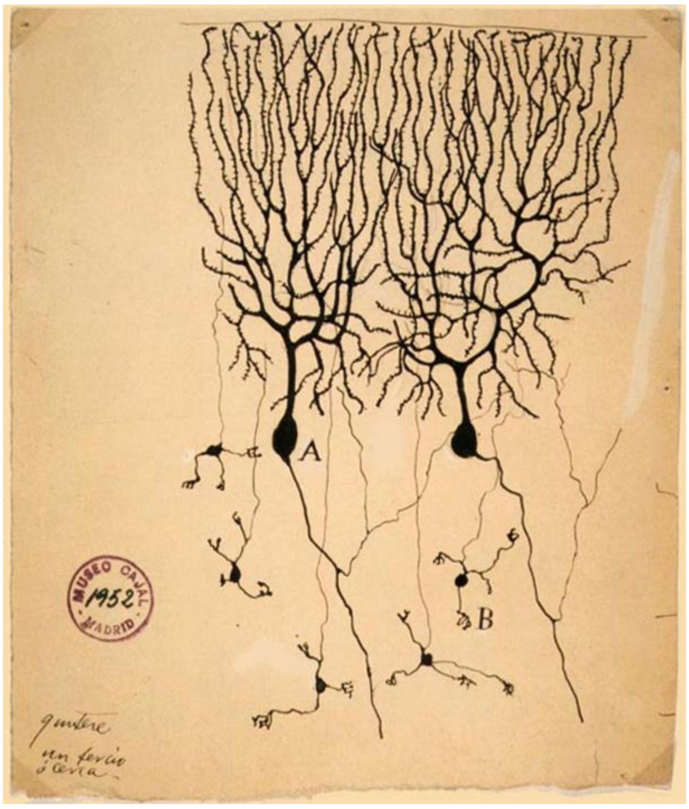
Most of all, I would like to thank the members of my family for their psychological and economical support during all the years of my studies.

Table of Contents

Table of Pictures

Chapter 1: Introduction

It took over a hundred years of research for scientists to result in what is considered to be the neuron structure as it is known today. The Spanish Nobel Prize laureate neuroanatomist, Santiago Ramón y Cajal [1], was the first who suggested that the neuron is the anatomical and functional unit of the nervous system. For his research about neural cells, he is regarded as the father of modern neuroscience.



Picture 1.1
Drawing of cerebellar Purkinje [2] cells by Santiago Ramón y Cajal

In the last decades, the scientific discoveries about the structure and function of the human brain, created needs for further research and study in this specific field. The scientists, in order to understand better how a brain manages and processes information, tried to create mathematical models, based on its functions.

The most important model is considered to be the one that was created by the biophysicists, Sir Andrew Fielding Huxley and Sir Alan Lloyd Hodgkin [3], in 1952. This mathematical model, which gave them the 1963 Nobel Prize in Physiology or Medicine, was close enough to the actual biological data. However, due to this fact, it was computationally expensive.

The rapid progress in the field of computers over the last twenty years, enabled the optimization of these mathematical models. Scientists could simulate a brain's function, faster and with the use of larger amount of data. These particular models, slightly modified, were also used as computational tools for fields like Machine Learning and Cognitive Science. It becomes clear that, as computers increase in speed and processing power, the existing models could be further optimized and newer, better models could be designed.

1.1 Thesis object

This diploma thesis aims at the searching of a way of displaying biological data.

Due to the fact that biologists who work in the neurobiology field are using standard forms for displaying exit data of a simulation, in this thesis a different approach is being tested. This approach constitutes an effort on the imaging of a non-photorealistic scene, based on scientific discoveries and microscopic photos.

In opposition with the current forms of displaying neuron data - such as tables, networks with nodes, graphs - a three-Dimensional space is visualized, by using the game engine Unity [4]. Three-dimensional biological models coexist with some of the forms of displaying, as mentioned above.

1.2 Contribution

In the contexts of this diploma thesis, an application has been designed in order to visualize the information obtained after a neural network simulation. The implementation of the Neural Network Simulation is detailed in the diploma thesis titled "Neural Network Simulation Speedup based on Reconfigurable Logic" [5]. The application also enables the input of simulation data, thus acting - in combination with the software system for simulation - as a comprehensive study tool and neural network representation.

The approach that is being followed for displaying the output data of a simulation, could easily be applicable as an educational tool for understanding the neurons' function. Being a relatively precise representation of a group of interconnected neurons, it gives the ability to someone to understand some of the neural system functions, even if this specific field is completely unknown to him.

1.3 Thesis structure

Coming to the end of the **first chapter**, a synoptic description of the content of the remaining chapters follows, as detailed below.

The **second chapter** contains a report of older related works and includes ways of visualization and biological data representation

The **third chapter** constitutes a full description of the project framework. It also includes an analytic description of the diploma thesis titled "Neural Network Simulation Speedup based on Reconfigurable Logic", to which this thesis was based on.

In the **fourth chapter**, the application is presented, describing the way that it is interfaced with the user.

In the **fifth chapter**, the application is analyzed, presenting information about the implementation.

The **sixth chapter** describes the way that the application was confirmed and evaluated.

In the **seventh chapter**, as a conclusion, a description of suggestions is contained for future work and possible extensions of this project.

Chapter 2: Related Work

The large amount of information accumulated over the years and the research on the human - or animal – brain, requires simulations and ways to represent the data. These are necessary both for scientists and non-specialized users, in order to understand better the functions and the way that the brain manages any information it acquires.

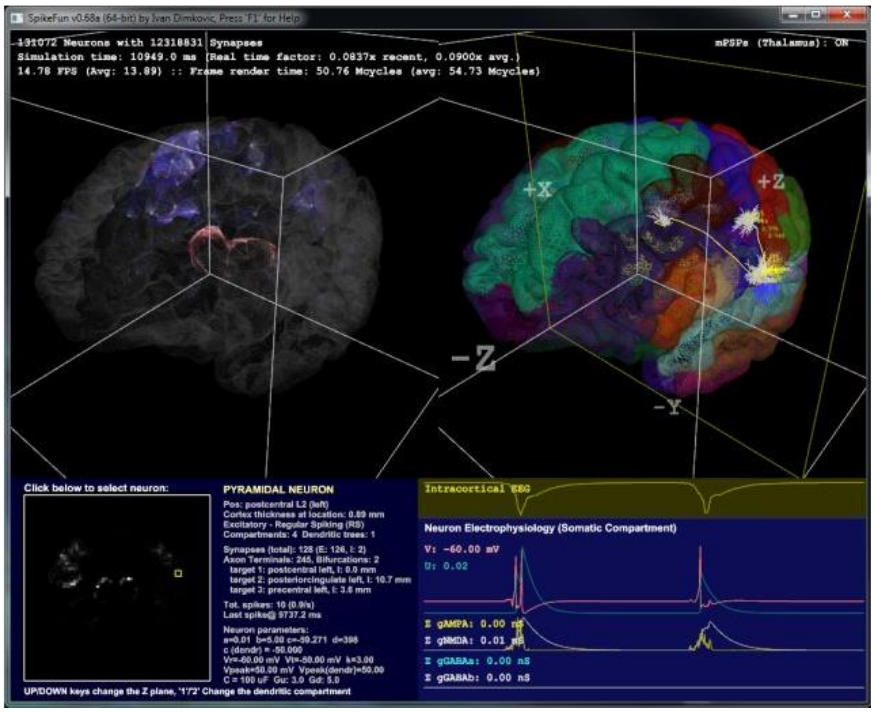
The Computational Biology Lab of IMBB/FORTH [6] has years of occupation on this object, in the contents of which software relating to the mapping cell structures and function of neural network models have been implemented. There is also important contribution of the Erasmus Brain Project [7], a multi-disciplinary consortium of university departments, research companies and institutes with a dual mission: To provide tools and methods for advanced brain research and to develop engineering solutions to specific neuroscience-related problems.

Furthermore, neural network simulations and visualizations have been designed by the University of Colorado Boulder [8] in order to be used as educational tools.

The research for the current Diploma Thesis is focused on three kinds of projects and applications:

- Those that are mainly related with the part of simulation and include UI for the input of parameters and / or a rudimentary data representation. Such applications are
- Those that are directly related to the part of visualization using UI and have a minimum interactivity with the user. Such applications are
- Visualizations with default parameters, without UI. Such applications are

DigiCortex [9] is a Biological Neural Network Simulator which is implementing Eugene Izhikevich's and Brette-Gerstner (AdEx) phenomenological models.


SpikeFun v0.68a (64-bit) by Ivan Dimkovic, Press 'F1' for Help
+X
+Z
-Z
-Y
Click below to select neuron:
Intracortical EEG

```
131072 Neurons with 12318831 Synapses                                   mPSPs (Thalamus): ON
Simulation time: 10949.0 ms (Real time factor: 0.0837x recent, 0.0900x avg.)
14.78 FPS (Avg: 13.89) :: Frame render time: 50.76 Mcycles (avg: 54.73 Mcycles)
```

PYRAMIDAL NEURON
Pos: postcentral L2 (left)
Cortex thickness at location: 0.89 mm
Excitatory - Regular Spiking (RS)
Compartments: 4 Dendritic trees: 1

Synapses (total): 128 (E: 126, I: 2)
Axon Terminals: 245, Bifurcations: 2
target 1: postcentral left, l: 0.0 mm
target 2: posteriorcingulate left, l: 10.7 mm
target 3: precentral left, l: 3.6 mm

Tot. spikes: 10 (0.9/s)
Last spike@ 9737.2 ms

Neuron parameters:
a=0.01 b=5.00 c=-59.271 d=398
c (dendr) = -50.000
Vr=-60.00 mV Vt=-50.00 mV k=3.00
Vpeak=50.00 mV Vpeak(dendr)=50.00
C = 100 uF Gu: 3.0 Gd: 5.0

UP/DOWN keys change the Z plane, '1'/'2' Change the dendritic compartment

Neuron Electrophysiology (Somatic Compartment)

```
V: -60.00 mV
U: 0.02
Σ gAMPA: 0.00 nS
Σ gNMDA: 0.01 nS
Σ gGABAa: 0.00 nS
Σ gGABAb: 0.00 nS
```

Picture 2.1
DigiCortex in operation "BrainView"

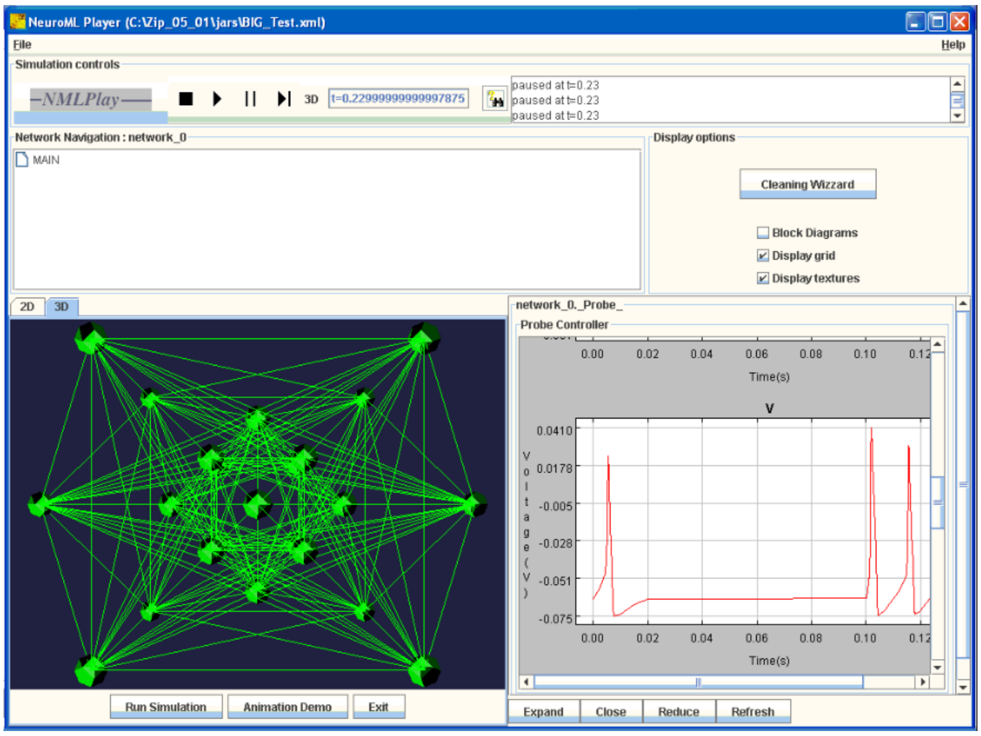
It enables the user to tweak individual parameters of the simulation such as number of neurons, modify the built-in neuron types, connectivity statistics (neuron synapse formation affinities, percentages of each post-synaptic neuron type) as well as various additional neuronal parameters (e.g. axonal action potential propagation velocities).

DigiCortex can simulate up to 16 million neurons and ~4 billion synapses. It includes AMPA, NMDA and GABA synaptic receptor kinetics as well as long-term and short-term synaptic plasticity.

Also, an OpenGL visualization of the simulation including real-time neuron picking, as well as FFT analysis of spiking statistics and synaptic weight tracking are contained in the Simulator.

DigiCortex is considered to be resource intensive - especially in terms of memory bandwidth. Current speed limits are mostly due to the memory I/O.

The NEOSIM project [10] is based on a parallel discrete event simulation kernel for running models of spiking neural networks on clusters of machines. It also includes a user interface for building and running simulations and a modules kit for extending the behavior of neurons and connectivity patterns.



t=0.22999999999997875

```
paused at t=0.23
paused at t=0.23
paused at t=0.23
```

Picture 2.2
NEOSIM's "Graphs and 3D visualization"

Real-Time Simulations of Synchronization in a Conductance-Based Neuronal Network with a Digital FPGA Hardware-Core[11] has been designed for real-time network simulations with up to 400 physiologically realistic, conductance-based neurons of the Hodgkin-Huxley type.

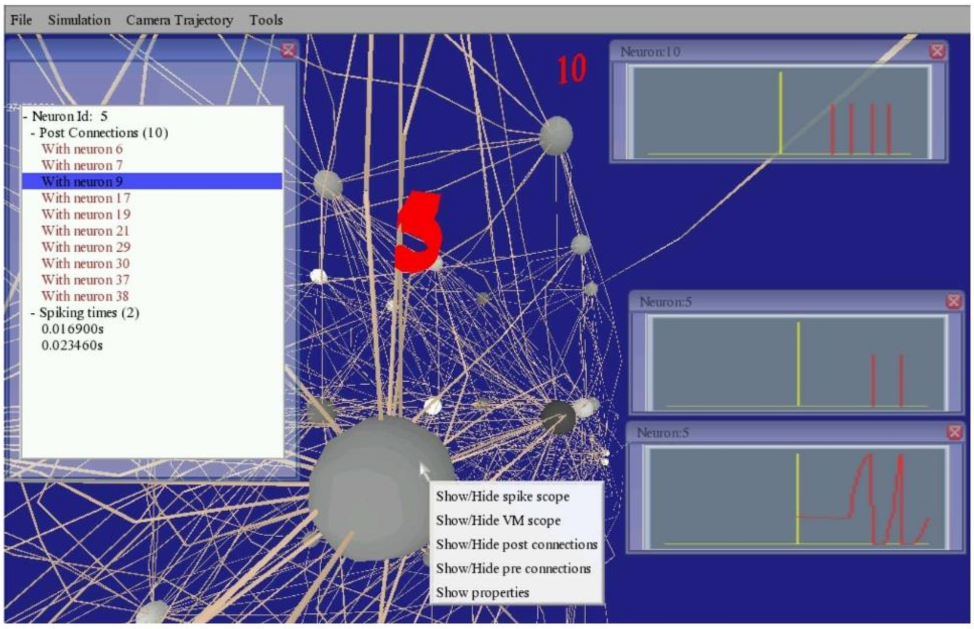
A PC-FPGA interface allows easy parameter adjustment and on-line display of basic synchronization measures like field potentials, spike times or color-coded voltages of the complete array.

SNN3DViewer [12] constitutes a visualization system dedicated to support analysis of dynamical processes in large spiking neural networks. The user can select any of the available applications for the design and simulation of spiking neural networks and to visualize the results using SNN3DViewer. It is able to render up to 100 000 interneuronal connections in the real-time.

As an application, offers an advanced graphical user interface, implemented with a CEGUI library [13], and designed according to the recommendations about the visual presentation clarity and easy graphical objects manipulation. The three-dimensional visualization is based on Direct3D graphic library [14]. Neurons are represented schematically as spheres. Each neuron is described by its 3D location, radius, firing times and optionally by the internal state trace (usually corresponding to the membrane potential of the biological neurons). The internal state Vm(t) is visualized by a neuron colour. The particular segments of the interneuronal connections are characterized by their strength and they are modelled as cylinders.

The application is equipped with a set of tools which additionally support the network analysis. These are e.g. graph windows which display the firing times or membrane potential of the selected neurons. Several graph windows can be presented simultaneously on the screen. The user can easily modify their parameters (position, size, etc).

In case of large networks, the application's performance is optimized by rendering only these connections that are in the observer camera visibility frustum.



Picture 2.3
Typical results of network visualization with SNN3DViewer

Towards the Visualization of Spiking Neurons in Virtual Reality[15] presents a prototype that addresses the visualization of the microscopic activity structure in the mammalian brain. It is trying to display the spiking behaviour of neurons in multiple layers based on large-scale simulations of the cortical microcircuit. Each one of these layers consists of 80.000 neurons and 0.3 billion synapses.

Excitatory cells are represented in a different way than the Inhibitory. The layers are outlined and an alternate coloring scheme is used for neurons in consecutive layers. Over time spiking neurons are highlighted and a global overview is provided by activating different layout types.

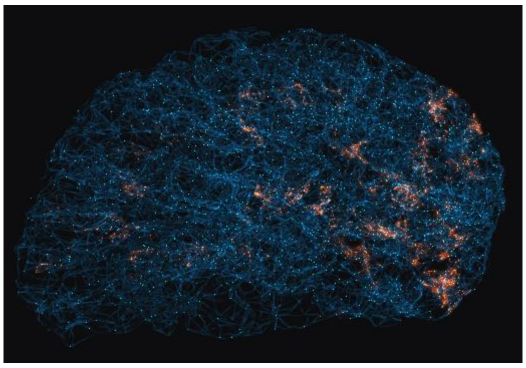
neuroConstruct[16] is a software application that facilitates the creation, visualization, and analysis of networks of multicompartmental neurons in 3D space. A graphical user interface allows model generation and modification without programming. Models within neuroConstruct are based on new simulator- independent NeuroML[17] standards, allowing automatic generation of code for NEURON[18] or GENESIS[19] simulators.

neuroConstruct was tested by reproducing published models and its simulator independence verified by comparing the same model on two simulators. It shows how more anatomically realistic network models can be created and their properties compared with experimental measurements by extending a published 1D cerebellar granule cell layer model to 3D.

The scale of simulations that can be run and visualized is therefore limited by the processor and video memory, respectively. Simulations of up to 5,000 multicompartmental neurons (50,000 simulated compartments) on a single processor with 2 GB of memory could be visualized with a 128 MB video card.

Except for the software applications that have already been described, there are some forms of visualization that are not based on simulation results or the input of certain parameters. These forms provide a more realistic visualization of a spiking neural network than the ones mentioned above.

(a) Neural Network Visualization by Reson8 [20]


(b) Neural Network Visualization by nxxcxx [21]

Picture 2.4
Neural Network Visualizations

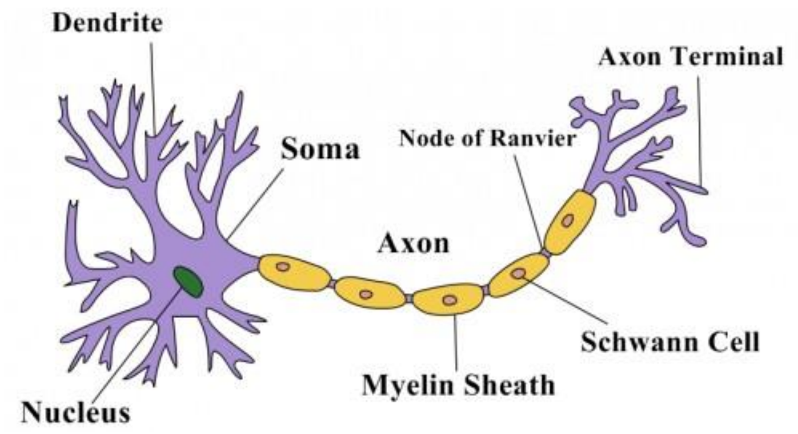
Chapter 3: Project Framework

3.1 Theoretical Background – The Nervous System

The Nervous System is the part of the animal's body that controls and coordinates - voluntarily and involuntarily- the functions of the organs. In vertebrate species it is divided in two main parts, the central nervous system (CNS) and the peripheral nervous system (PNS). The CNS - consists of brain and spinal cord – is responsible for the association and integration of neural information, whilst, PNS consists mostly of nerves that transmit information from the body to the CNS and vice versa.

On a cellular level, the nervous system is defined by the presence of two types of cell, the neuron and the glial cell.

While, glial cells provide structural and metabolic support, a neuron is considered to be the main structural and functional unit of the nervous system. As a cell, is responsible for processing and transmitting information using electrical and chemical signals.



Picture 3.1
The structure of a multi-polar neuron

Despite the structural classification based on polarity, the multi-polar neurons constitute the majority of neurons, mainly in the brain. Soma, dendrites, axon and axon terminals are the main parts of a neuron cell. Each one of these parts contributes with a specific way to cell's function.

The **soma** is the body of the neuron. As it contains the nucleus, most protein synthesis occurs here.

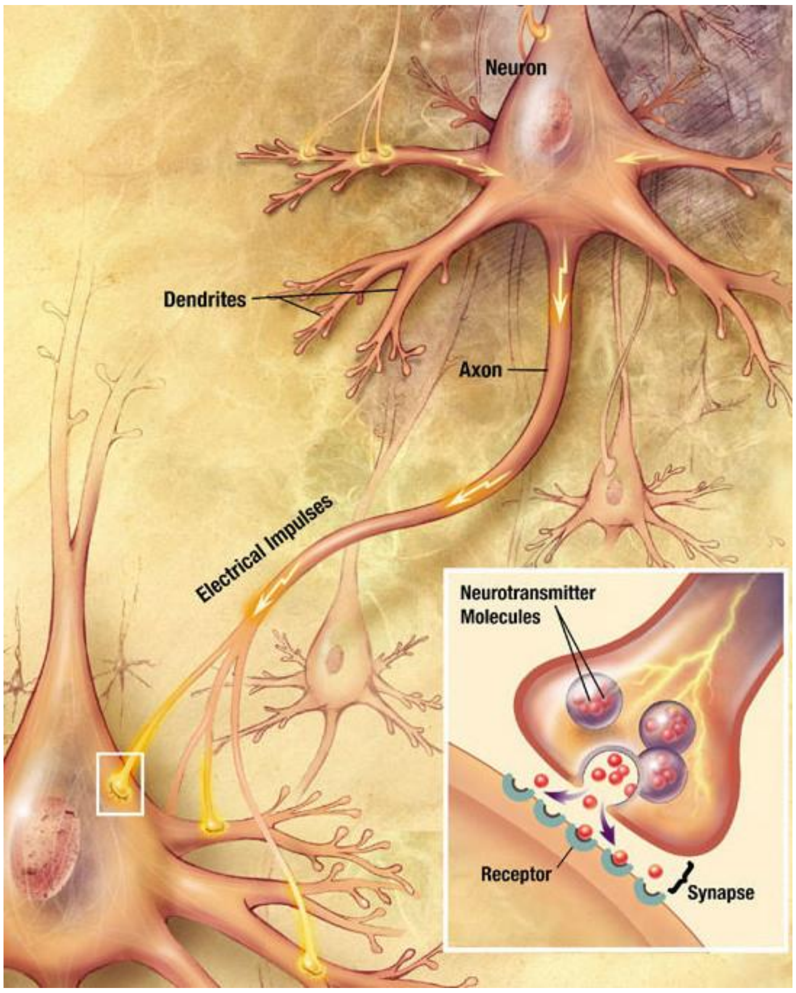
The **dendrites** of a neuron are cellular extensions with many branches. This overall shape and structure is referred to metaphorically as a dendritic tree. This is where the majority of input to the neuron occurs via the dendritic spine.

The **axon** is a finer, cable-like projection that carries nerve signals away from the soma (and also carries some types of information back to it).

The **axon terminal** contains specialized structures where neurotransmitter chemicals are released to communicate with target neurons.

Neuron cells have two important properties, conductivity and excitability, which enables them to connect with each other.

An infinitesimal vacuum between one neuron's axon terminals (presynaptic neuron) and another neuron's dendrite -or soma (postsynaptic neuron), called a **synapse**, permits a neuron to pass an electrical or chemical signal to another neuron.



Picture 3.2
The connection between two neurons -
a synapse in the bottom right corner

Synaptic transmission is called a chemical process, during of which a chemical neurotransmitter is released from the presynaptic neuron and activates specific proteins in the postsynaptic neuron. This process enables the transmission of information between two neurons and can excite or inhibit the postsynaptic neuron.

Chemical synapses can be classified according to the kind of neurotransmitter that is being released (**kind**). **AMPA** and **NMDA** types have excitatory character, in contrast with **GABAa** and **GABAb** types that try to inhibit postsynaptic neuron's activity.

A factor called **synaptic weight** sets the amount of electrical activity that will be transmitted by the synapse. This factor depends on synapse's width and the distance between presynaptic and postsynaptic neurons.

The process that was described above takes place when an electrical signal reaches the axon terminal (–). In this case the neuron “fires”. This electrical signal called action potential of a neuron (measured in millivolts), is the voltage pulse generated when the neuron is stimulated electrically (by a presynaptic neuron or the environment – **external stimuli**). The electrical signal is also known as spike and therefore, networks that consist of connected neurons are known as Spiking Neural Networks (SNNs). These signals are generated and propagated by charge-carrying ions including sodium (Na^+), potassium (K^+), chloride (Cl^-), and calcium (Ca^{2+}).

A neuron cell controls in a two level way if the spike will be transferred to its axon terminals. Initially, each one of its dendrites adds the incoming – at a specific time - electrical signal from the synapses (positive and negative sign depending on the synapse's kind). If the total sum has reached or has exceeded a specific voltage value (called **firing threshold**), the dendrite produces a spike and sends the electric potential in the soma. Otherwise, the dendrite is considered to be inactive and its potential is lost.

In dendrite's layer, the firing lasts for 50 milliseconds. Only the inhibitory kind of synapses can contribute to the electrical potential during this period of time. So, in case that dendrite has "fired", they can set it closer to or even below the threshold level. Firing threshold differs for each dendrite.

In neuron's layer, soma adds the electrical signals that are transmitted from its dendrites. If the total sum exceeds soma's threshold voltage value, the neuron "fires" and propagates an action potential in the range of around 100 mV to the axon and eventually to the axon terminal (lasts for 1 millisecond). Otherwise, the neuron remains at rest. A neuron could also act – as well as the synapses – in an exciting of inhibiting (**kind**) way, affecting the postsynaptic neurons that it is connected with. Firing threshold differs for each neuron.

The number of neurons, according to array tomography, is proven to be -on average- about 86 billion in the adult human brain and each one of them is connected to a number of between 5,000 and 200,000 other neurons.

3.2 Implementation of the Simulator

In the contexts of the diploma thesis titled "Neural Network Simulation Speedup based on Reconfigurable Logic", an emulator for a generic and detailed spiking neural network based on the analysis of the Hodgkin and Huxley model has been implemented. This spiking neural network model can simulate a maximum number of 70 neurons, with 64 dendrites per neuron and 512 synapses per dendrite with partial connections.

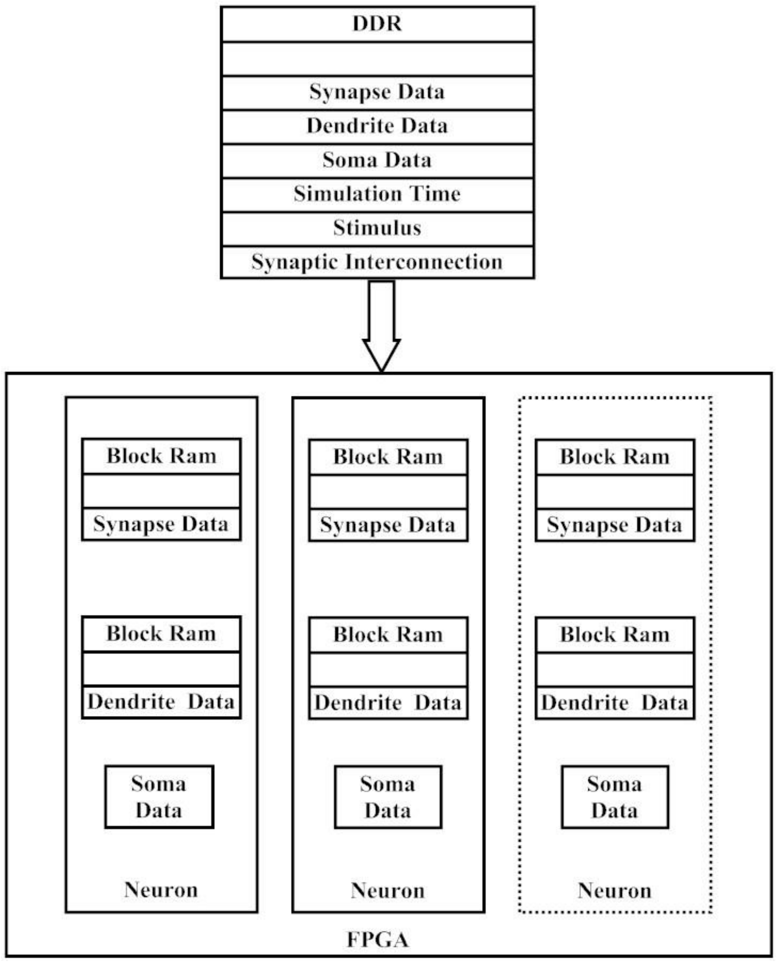
The architecture was implemented on the Convey HC-2ex hybrid FPGA super-computer [22], because of the large amount of data that should be stored in external memory and the required memory bandwidth. The Convey HC-2ex has 4 Virtex-6 LX760 FPGAs. All FPGAs share the same memory and a coprocessor. Also, the platform has 14 memory controllers for parallel read. This implementation was integrated into a single FPGA and operates only one memory controller when reading data from the memory, and two memory controllers when storing the results in memory.

The system implementation is divided into 5 basic functions, the initialization of the synapses, the initialization of dendrites, the update of synapse state, the computation of neuron state and the data storage.

The synapse data, the dendrite data, the neuron interconnection, the external stimuli and the simulation time are all initially stored in an external memory of the hybrid FPGA platform Convey HC-2ex. The data that are read from the external memory of the platform have the maximum size of 64bits. The lower 32 bits are assigned to data and the upper 32 bits to addresses.

At the beginning of each simulation, information of each type of synapse and each dendrite are stored in the internal memory of the FPGA. Each neuron has a dual port Block RAM (BRAM) of 8192 lines depth and 26bits width and a single port BRAM of 64 lines depth and 16bits width that can keep all the synapse and dendrite data of the neuron, respectively.

The synapses are initialized with the parameters that characterize them such as the kind of synapse, the synaptic weight, the response counter and synapse state. Each block of 27bits width keeps the data of a synapse. The dendrites are initialized with the dendrite threshold. If a synapse or a dendrite doesn't take values, it remains inactive and does not affect the system.

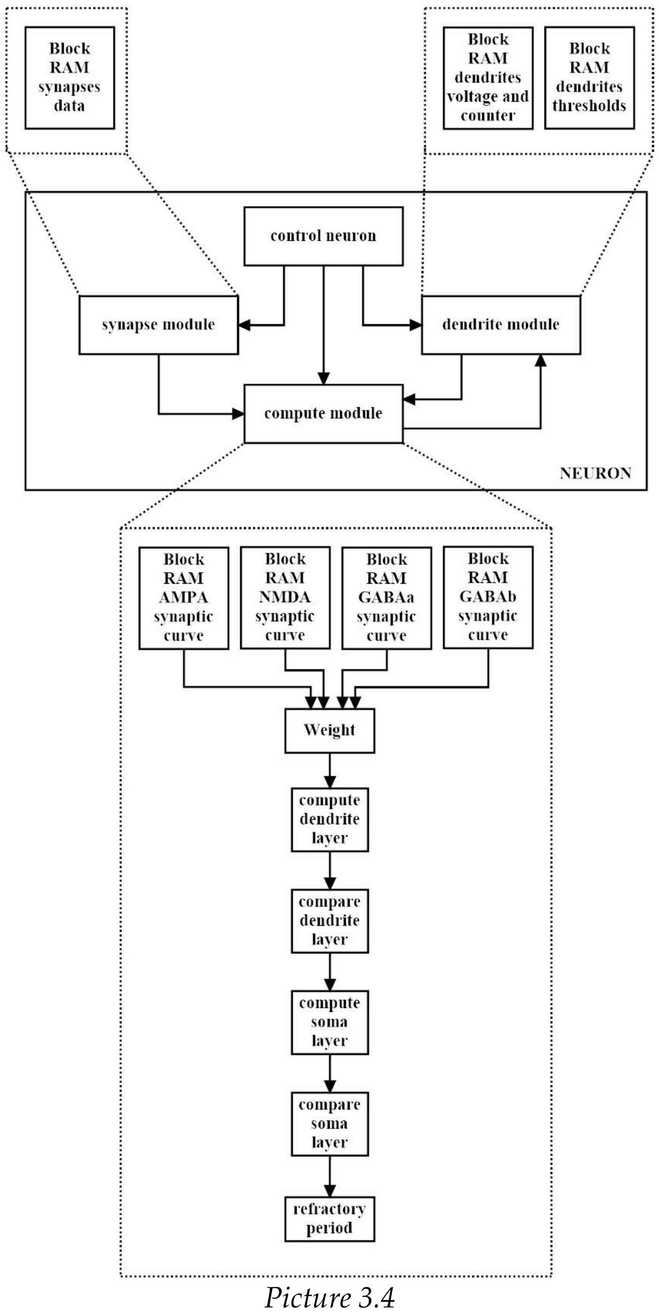


Picture 3.3
The distribution of data
in internal memory (BRAM) & external memory (DDR)

The update of synapse state is iterated in each simulation time step. During this function, the synapses are activated, unless the neuron which is connected with them has triggered a "fire". The system has a vector that the "fires" of the neurons are only stored.

The computation neuron state is performed to calculate the action potential in each simulation time step. Each neuron has four additional single block BRAMs that store the characteristic synaptic curves of each kind of synapses. Each synaptic curve has different duration, so the depth of each BRAM differs from the others. Also, each neuron has a 64 depth and 16 width BRAM which keeps the dendritic voltage for about 50ms simulation time.

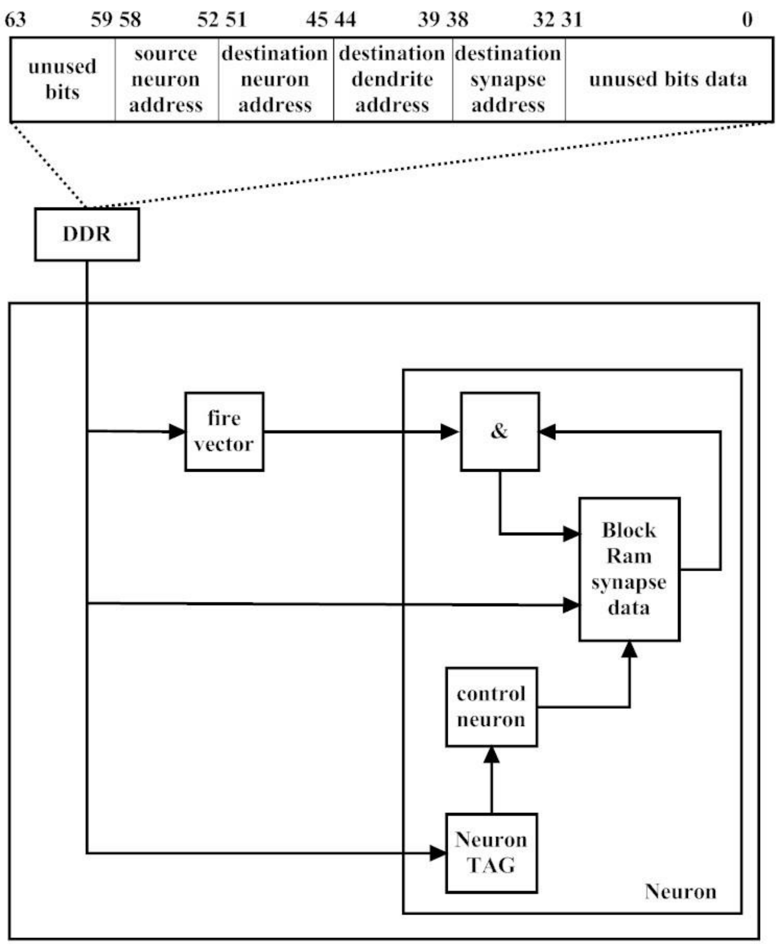
At the start of the calculation, all neurons are evaluated in parallel, and they serially read the synapse data from their BRAMs. In case of the synapse is enabled then it samples the discrete values of the characteristic synaptic curve. The discrete values are divided with the weight and the result summed in the dendrite layer. When all synapses of the dendrite are computed, the total voltage of dendrite is compared with the dendrite threshold. If this value is greater than the threshold, it will go to the soma layer and be stored in a dendrite voltage BRAM for about 50ms.



Picture 3.4
Neuron Architecture

When all dendrites are computed, the soma checks if the overall voltage has exceeded the soma threshold to trigger a fire or not. The computation of the neurons is fully pipelined. When the process is finished, the array with the neuron firings is updated. In the data storage function, the neuron state and the soma voltage are stored in the external memory.

A complete simulation cycle is completed when the interconnection update, the computation neuron and the data storage have finished. Each neuron has a control unit that organizes, controls and synchronizes the functions. In addition, the system has a general control unit that synchronize all the neuron units and the external memory.



Picture 3.5
Update of synapse state Architecture

Finally, as a result, it was calculated that the FPGA implementation of the network on a single Virtex-6 LX760 FPGA provides a speedup of 24-40X times of similar neuron networks simulations that are executed in software on a CPU with 4 cores at 3.10GHz.

3.3 Implementation Layout

With the simulation algorithm already implemented, as it was analyzed in the previous subchapter, the original aim of this diploma thesis was to find a way of presenting the exported data from the simulation.

Specifically, the presentation of the results (output data) requires the displaying of the extracted information of the simulation in a manner both comprehensive and complete. Basically, what the user eventually sees, should be the output in its entirety without creating information overload at the same time. The results should be presented to the user in a way that is easy to understand, without the difficulty to be read and without the user being confused by their volume.

Initially an implementation was chosen so that is not based on standard forms of displaying exit data of a simulation such as raster plot, due to the fact that in that case the results are presented in their entirety all at once. On the contrary, a method was searched in order to achieve a variation depending on the time visualization of the data, so that there are distinct visualization periods, each one of which corresponding to a specific step of the simulation.

The first idea in order to implement the method mentioned above was to find a way the output data to be provided to the user with a continuous flow. For example, the results are visualized per some steps without the completion of the simulation being necessary. That could be implemented either by acquiring output data per simulation step or at the end of a certain number of steps, for example a hundred.

This specific idea was dismissed immediately as the output results are exported faster than it is possible to be displayed in a continuous flow, let alone for the human eye to perceive them. Additionally, the time required in order the simulation to be completed is of the order of seconds, so it is not prohibitive for the user to wait for its completion.

Having decided that the visualization of the results will start at the end of the simulation and after the data have been totally exported, the issue that had to be addressed subsequently, was the way in which neurons and dendrites will be represented in a continuous flow visualization.

For the presentation of the cells, the imprinting of the neurons and dendrites was initially tested by using the Open GL in forms of spheres or circles which changed color in the presence of potential that exceeded the threshold - firing. This plan although it presented the largest part of the information in a satisfactory manner, had shortcomings both in details of cell function and aesthetic.

For example, the use of the implementation that was mentioned before can be fully understood by an expert in this particular scientific field. However, for someone not so familiar with the field of neuroscience, a more plausible cell model or the representation of processes such as signal transduction from the neuron's soma to the axes would help in better understanding and learning of the functions of a neural network.

The use of three-dimensional cell models covered the expectations set previously. However, given the fact that the simulation algorithm was designed to implement a network of a maximum of sixty neurons and sixty four dendrites per neuron, this implementation had to meet these requirements.

Moreover, in order the application to be complete, an addition of tools widely known for representing simulation neuronal network data like plot and raster plot, was selected. These additions were adjusted to this specific design, driven by the effort the information that the user receives to be as complete as possible.

As a secondary aim of this diploma thesis was addressed the design of a user-friendly way of input data necessary for the simulation. The insertion of the data (input data) requires a way of implementation that can be understood and be used by people without programming skills and minimum comprehension of neuroscience.

Based on what should be implemented as described hereinabove, the game engine Unity was chosen mainly for the possibilities offered and the ease of use in the implementation of the visualization. The fact that Unity provides its own serviceable UI, made it the most suitable to meet the needs of the insertion data design and more.

3.4 Implementation Platform

Game Engine Unity was used to develop a standalone application for Microsoft Windows. Unity is being used to develop video games for PC, consoles, mobile devices and websites for over ten years. C# was used as scripting programming language. C# first appeared in 2000 and it continues to be used as a general-purpose, object-oriented programming language.

Chapter 4: User Interface (UI)

In the contexts of this diploma thesis, an application has been designed, as it has already been indicated. This application, as a Windows Standalone one, doesn't require no installation for someone to use it, just to run an .exe file.

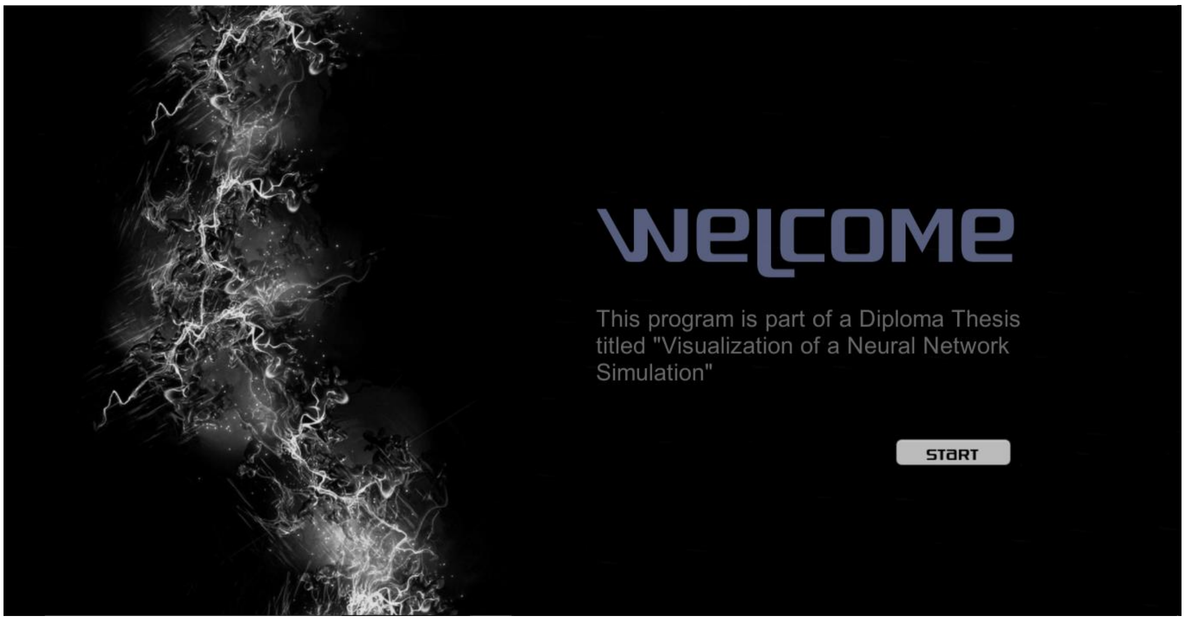
Picture 4.1
Run the .exe file

The interacting of the user with the application as well as the structure of the implementation, consist of three parts. In the first part, the input of the simulation's parameters takes place. Conversely, the last two parts concern the management of the exit data.

In this chapter, the implemented application will be analyzed only on the level that the user is interfacing with it - like a user manual. A detailed description of the implementation for each part is included in the next chapter.

4.1 Input Data (UI)

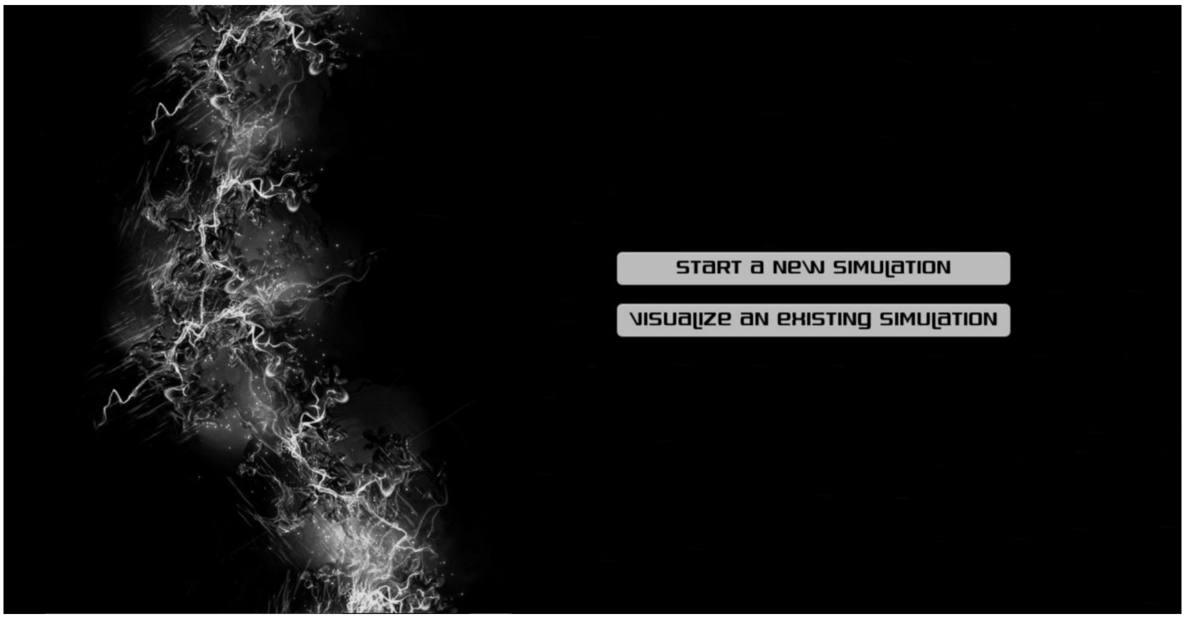
Initially, a Welcome Page welcomes the user to the program that he is about to use and informs him about it. A button Start is also contained, which must be pressed in order the user to continue.



This program is part of a Diploma Thesis titled "Visualization of a Neural Network Simulation"

Picture 4.2
Welcome Page – step one

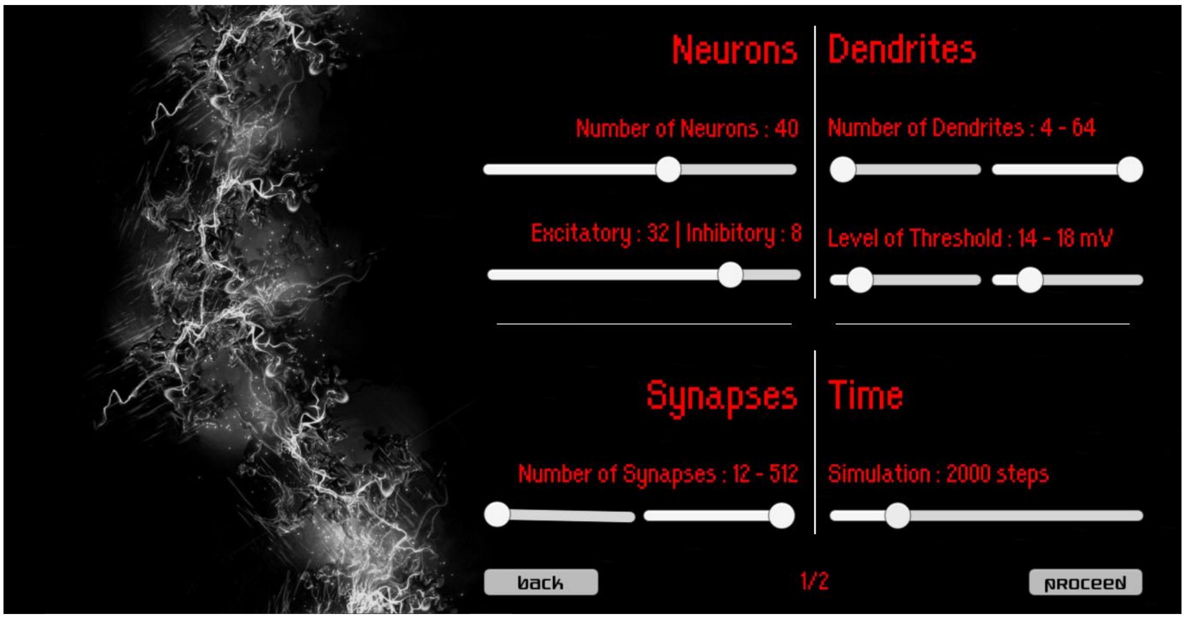
In the very next page, the user has to choose for the first time between two options. The first choice, enables him to determine every parameter and run a simulation of the neural network. On the other hand, by picking the second choice, the user is given the ability to load the exit data of an existing simulation.



Picture 4.3
Form of Action Choice – step two

The next window that will be appeared to the user is based on the choice that was made.

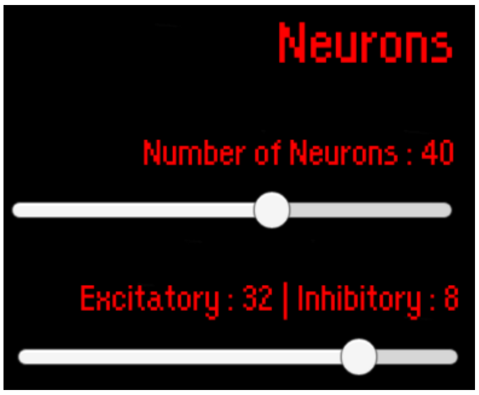
In the first case, an Input of Parameters panel comes out – one out of two. As it will be described below in this step the user customized parameters related to neurons, dendrites, synapses and the duration of the simulation time.



Number of Neurons : 40
Excitatory : 32 | Inhibitory : 8
Number of Dendrites : 4 - 64
Level of Threshold : 14 - 18 mV
Number of Synapses : 12 - 512
Simulation : 2000 steps

Picture 4.4
Input of Parameters Panel (1) – step three

Neurons



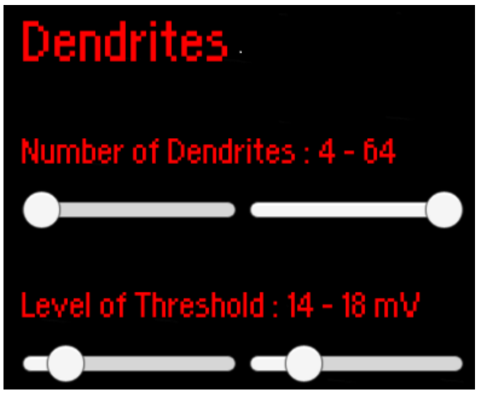
Picture 4.4.1
Neurons Parameters

The upper-left quarter-panel contains two Sliders.

By moving the "Number of Neurons" slider left and right the user chooses the amount of neurons between a minimum of ten(10) and a maximum of sixty(60).

By moving the second slider left and right the user chooses the amount of excitatory and inhibitory neurons. The default percentage is 80% Excitatory and 20% Inhibitory.

Dendrites



Picture 4.4.2
Dendrites Parameters

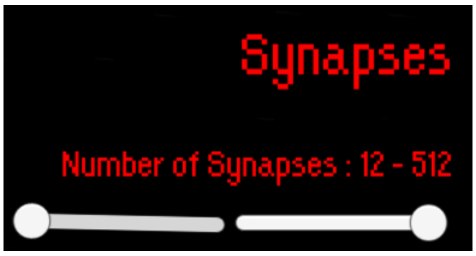
The upper-right quarter-panel also contains four sliders in two pairs.

The "Number of Dendrites" sliders adjust the minimum of four (4) - and the maximum – of sixty four (64) – amount of dendrites a neuron can contain. The number for each neuron is determined using uniform distribution between the minimum and the maximum number of dendrites.

In case that the user desires a specific amount of dendrites for each neuron, he can just set both of the sliders on the same value.

The "Level of Threshold" sliders function the same way. The level of threshold for each dendrite is determined using uniform distribution - between five (5) and seventy (70) millivolts.

Synapses



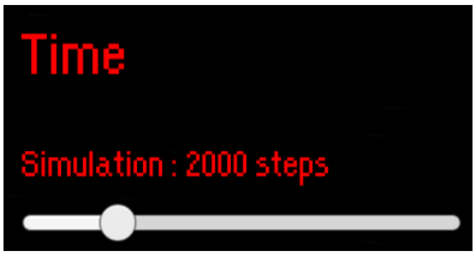
Picture 4.4.3
Synapses Parameters

The down-left quarter-panel contains two slider in one pair.

The "Number of Synapses" sliders adjust the minimum and the maximum amount of synapses a dendrite can contain. The number of synapses for each dendrite is determined using uniform distribution between a minimum of twelve (12) and a maximum value of five hundred twelve (512).

Like in the Dendrites quarter panel if the sliders coincide, each dendrite has this specific amount of synapses.

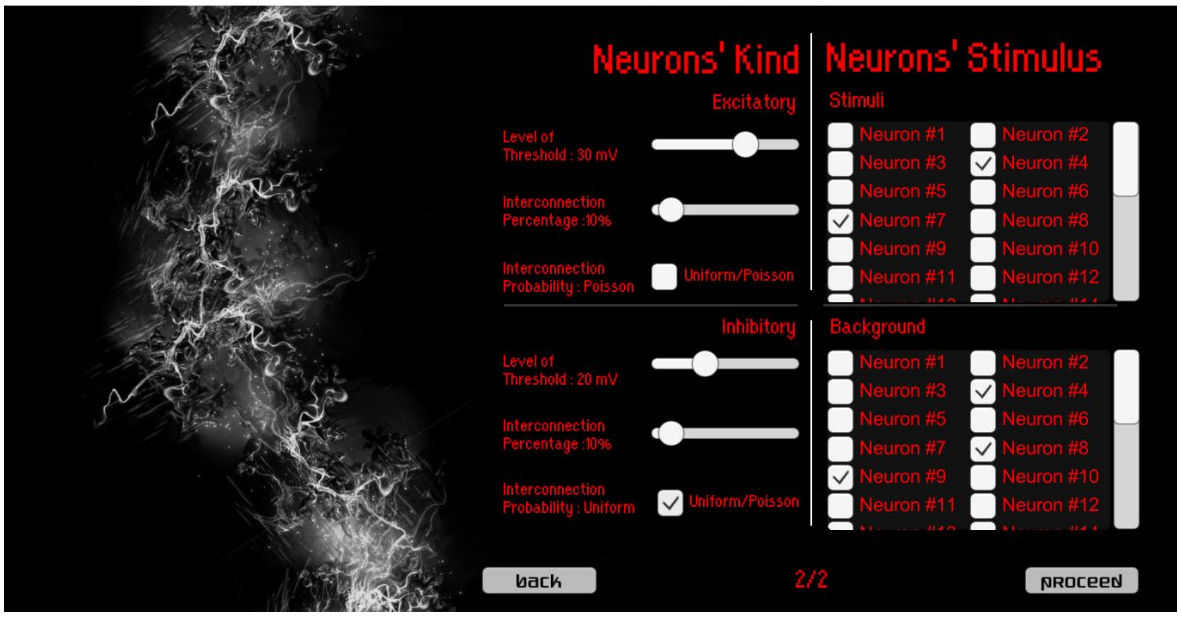
Time



Picture 4.4.4
Simulation Time Parameters

Finally, the quarter-panel in the down-right corner contains a single-slider to adjust the duration of the simulation time in steps. It is worth mentioning that a simulation step equals to a zero point one (0.1) seconds.

The first Input of Parameters also includes two buttons. Pressing the Back button leads the user to the Form of Action Page. By picking the Proceed button another Input of Parameters Panel (the second one out of the two) appears.



Neurons' Kind

Excitatory

Level of Threshold : 30 mV

Interconnection Percentage :10%

Interconnection Probability : Poisson ☐ Uniform/Poisson

Inhibitory

Level of Threshold : 20 mV

Interconnection Percentage :10%

Interconnection Probability : Uniform ✓ Uniform/Poisson

Neurons' Stimulus

Stimuli

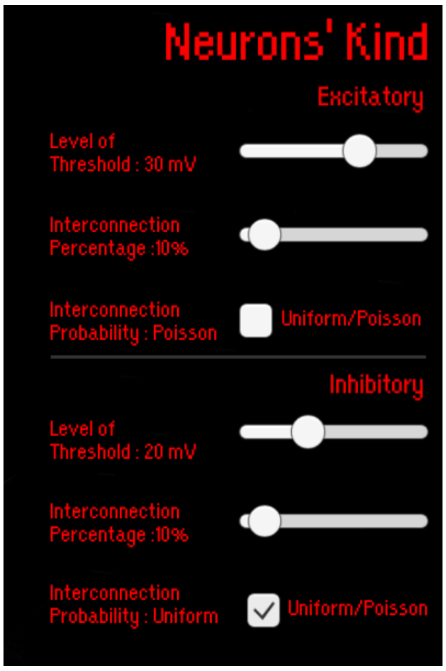
☐ Neuron #1	☐ Neuron #2
☐ Neuron #3	✓ Neuron #4
☐ Neuron #5	☐ Neuron #6
✓ Neuron #7	☐ Neuron #8
☐ Neuron #9	☐ Neuron #10
☐ Neuron #11	☐ Neuron #12

Background

☐ Neuron #1	☐ Neuron #2
☐ Neuron #3	✓ Neuron #4
☐ Neuron #5	☐ Neuron #6
☐ Neuron #7	✓ Neuron #8
✓ Neuron #9	☐ Neuron #10
☐ Neuron #11	☐ Neuron #12

Picture 4.5
Input of Parameters Panel (2) – step four

Kind



Neurons' Kind

Excitatory

Level of Threshold : 30 mV

Interconnection Percentage :10%

Interconnection Probability : Poisson ☐ Uniform/Poisson

Inhibitory

Level of Threshold : 20 mV

Interconnection Percentage :10%

Interconnection Probability : Uniform ✓ Uniform/Poisson

Picture 4.5.1
Kind Parameters

The half-panel on the left enables the user to customize parameters related to the neurons' kind.

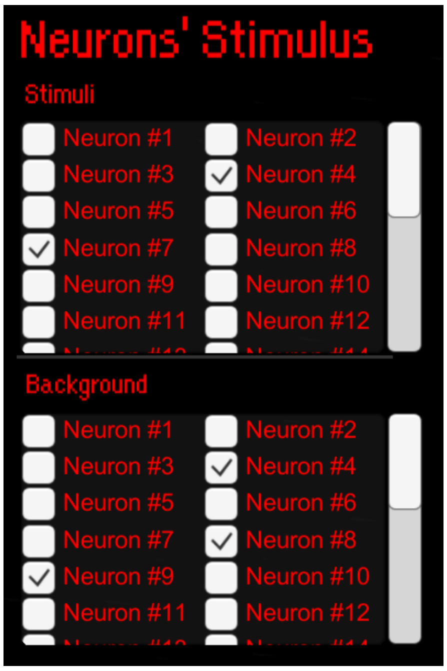
Each neuron can be either excitatory or inhibitory. For each kind parameters are determined separately.

By moving the "Level of Threshold" slider left and right the user imports the soma's threshold in mV. The range varies between ten (10) millivolts and forty (40) millivolts.

The "Interconnection Percentage" slider is used to set the percentage (between 5% and 100%) of the rest of the neurons to which each one of the neurons' signal is transmitted.

The user can also define the distribution that the Interconnection Percentage is based on by clicking on the "Interconnection Probability" checkbox. Checked stands for Uniform distribution whilst unchecked for Poisson. In case of picking Poisson distribution a neuron has more possibilities to be connected to one of these that are nearest to it rather than a distant one.

Stimulus



Picture 4.5.2
Stimulus Parameters

The half-panel on the right enables the user to customize parameters related to the neurons' stimulus.

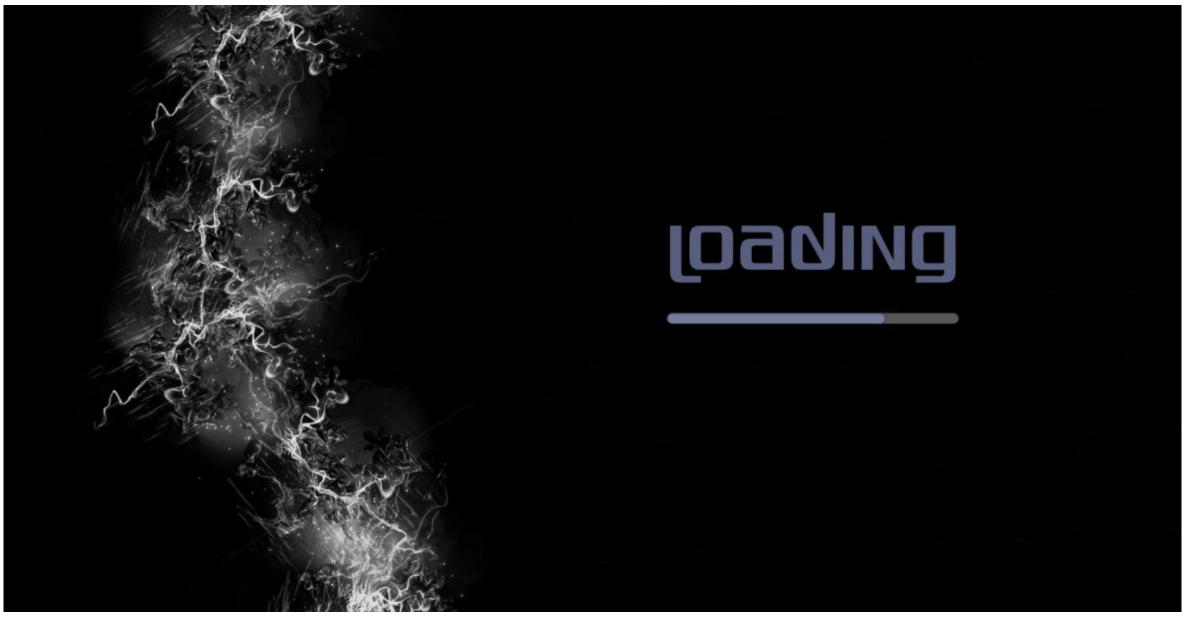
It contains two lists of neurons. The first one is referred to the neurons that have an external stimulus at the start of the simulation (t=0).

The second list is about the neurons that will receive an extra potential at specific times during the simulation.

User has the ability to check neurons that he desires to have the stimuli or to receive the extra potential.

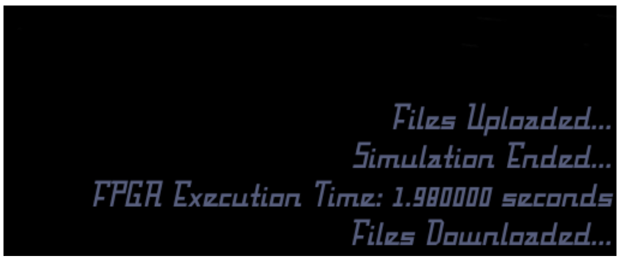
For every list a default percentage of neurons (set to 20%) has been randomly selected to be checked.

The second Input of Parameters also includes two buttons. The pressing of the Back button leads the user to the Input of Parameters Panel (1). By picking the Proceed button the settings of the simulation are completed and a Loading Page appears.



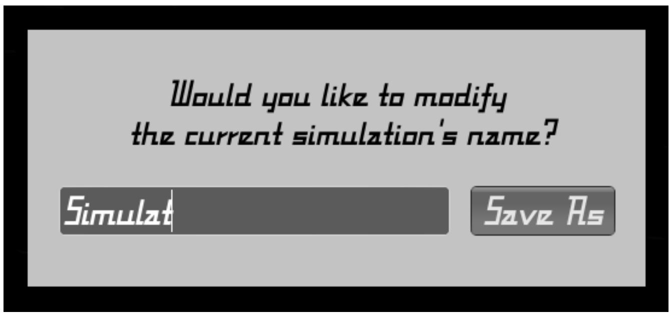
Picture 4.6
Loading Page – step five

As the gray bar is loading the user is updated on the procedures that are taking place in the background (files uploading to the serve, simulation running) - as well as the FPGA's Simulation time.


Files Uploaded...
Simulation Ended...
FPGA Execution Time: 1.980000 seconds
Files Downloaded...

Picture 4.7
Message Board Pop-up Window

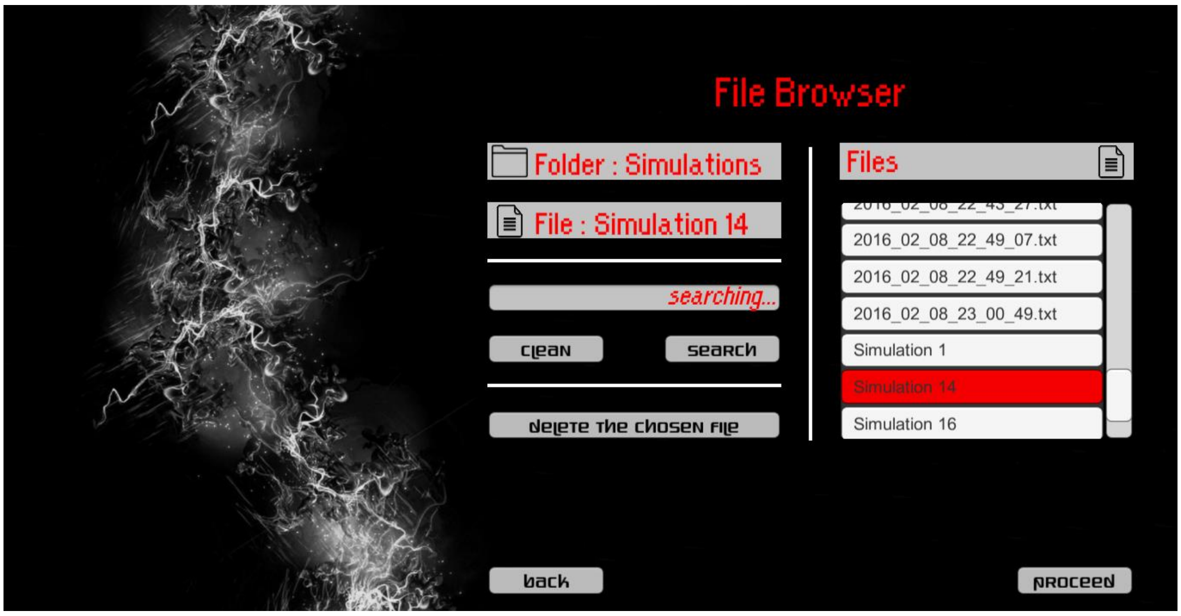
When all the actions have been completed, a pop-up window appears enabling the user to name the file of the current simulation. In case that the Input field was not modified the current simulation file's name will be the date and time that it took place.


Would you like to modify
the current simulation's name?
Simulat
Save As

Picture 4.8
Save Simulation Pop-up Window

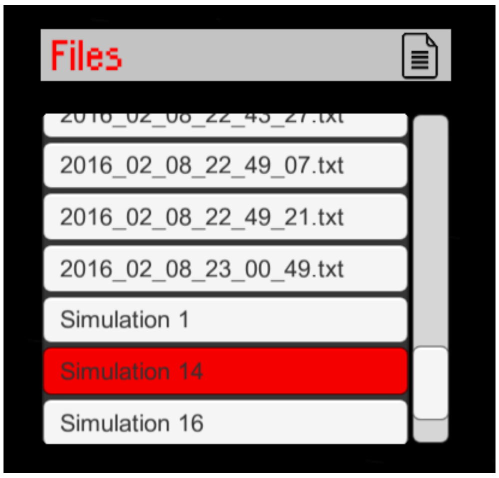
After saving the file, the user is lead to the final step of the “Input Data” part, which is the choice of the visualization type he desires to display. This final step is described in the end of the sub-chapter.

If the second choice in the Form of Action Page is picked, a File Browser panel appears.



Picture 4.9
File Browser Panel – step six

List of Files



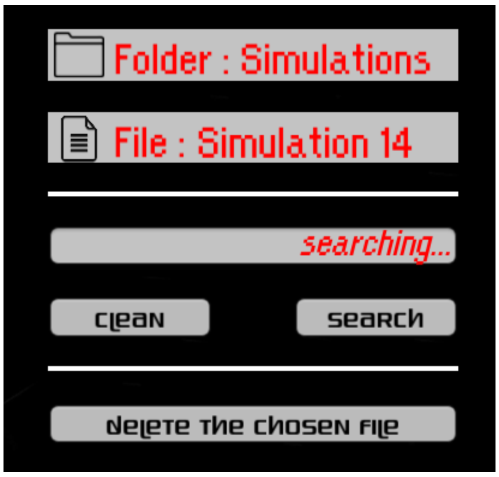
Picture 4.9.1
List of Simulation Files

The right half-panel contains a list of files. Each one of them represents an existing - ran in the past - simulation.

The names of the files are determined by the user, as he named them in the part that was described earlier.

When a file is picked, it is highlighted with red color.

List of Actions



Picture 4.9.2
List of Actions

The left half-panel consists of three distinctly parts. The upper one, informs the user if he has chosen a file from the right list by displaying its name in the "File" Text.

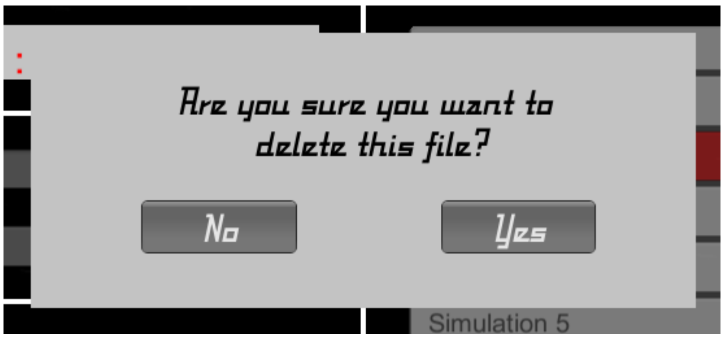
The second contains an input field and two buttons. The user can search for a specific file by importing a pattern in the input field and pressing the Search button. Then the files that contain the pattern appear in the right-half list.

By pressing the Clean button all the files are displayed, as they are not filtered by any pattern.

Also, in case that the user has picked a file, its name in the "File" Text will be erased.

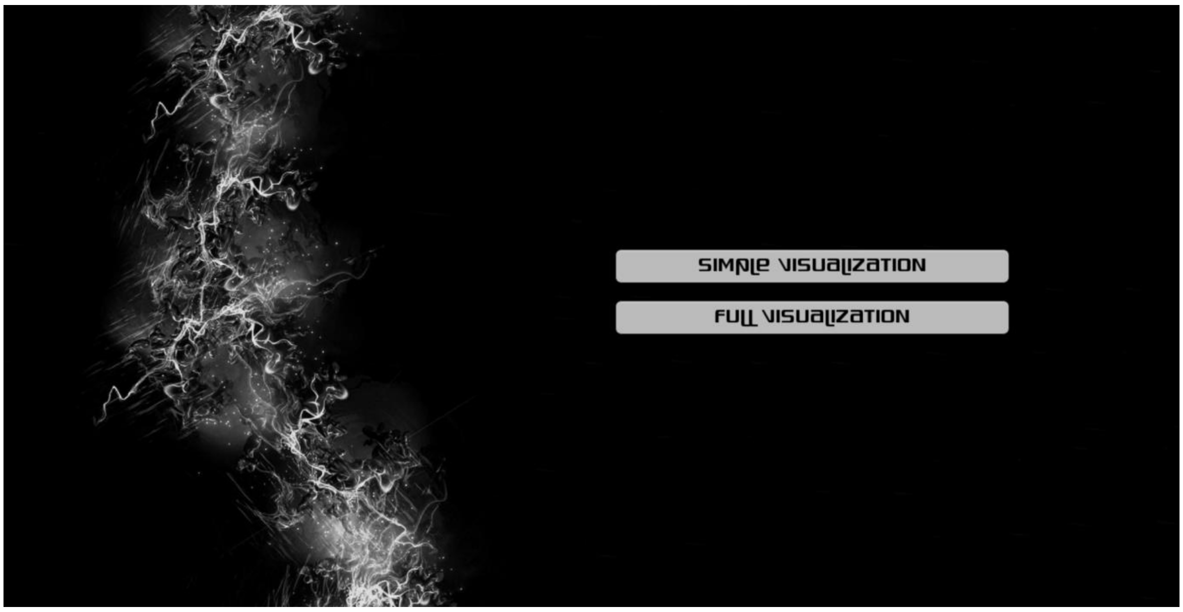
The half-panel also included a Delete button. In order to be interactable, a file has to be chosen.

In order to delete one of the simulation files, a pop-up window asks from the user to confirm his decision. In case of confirmation, the selected simulation cannot be recovered.



Picture 4.10
Delete File Pop-up Window

The File Browser also includes two buttons. Pressing the Back button leads the user to the Form of Action Page. By picking the Proceed button another – interactable only in case that a file has been selected – the user is leaded in the final step of the "Input Data" part, which is the choice of the visualization type he desires to display.



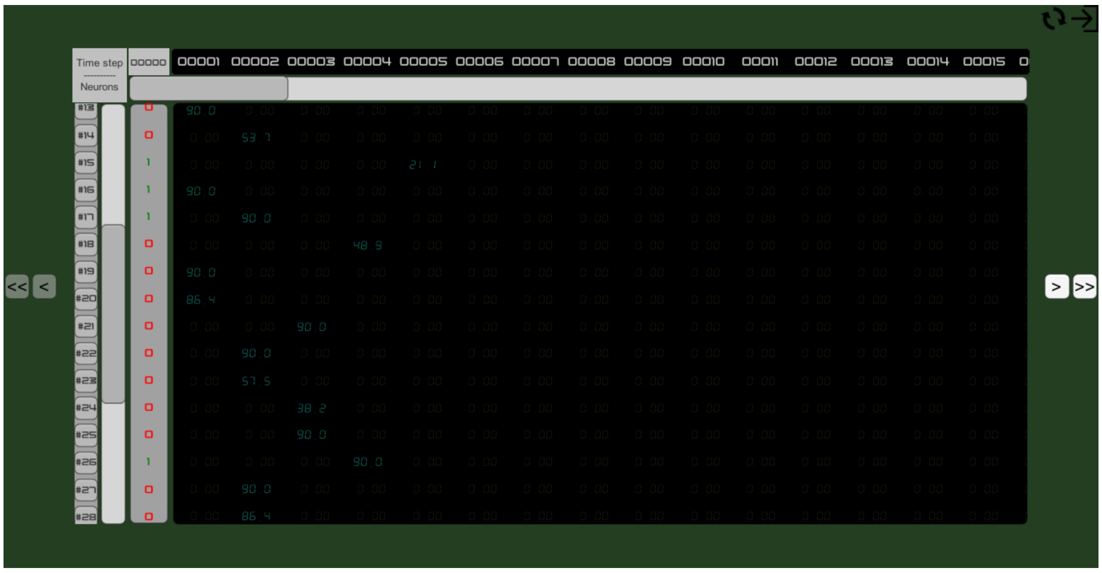
Picture 4.11
Form of Visualization Choice – step seven

At the end of the simulation, or after the loading of an existing one, the user has the ability to choose between two kinds of displaying the exit data. Each one will be described separately later on.

4.2 Two-Dimensional Visualization (UI)

The first form of visualization, is considered to be the simplest one out of the two kinds of displaying simulation's exit data. It is based on the idea of raster-plot, but it is a little bit more complicated and more analytic.

The user, in a first plane, sees a table that contains the action potential of each neuron for every step of the simulation. The table can be divided in three regions.

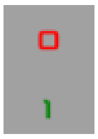


Time step / Neurons	00000	00001	00002	00003	00004	00005	00006	00007	00008	00009	00010	00011	00012	00013	00014	00015
#13	0	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#14	0	0.00	53.7	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#15	1	0.00	0.00	0.00	0.00	21.1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#16	1	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#17	1	0.00	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#18	0	0.00	0.00	0.00	48.9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#19	0	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#20	0	86.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#21	0	0.00	0.00	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#22	0	0.00	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#23	0	0.00	57.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#24	0	0.00	0.00	38.2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#25	0	0.00	0.00	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#26	1	0.00	0.00	0.00	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#27	0	0.00	90.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
#28	0	0.00	86.4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Picture 4.12
Table of Neurons' Action Potential

The first one in the left of the table is neuron related. It contains a vertical scrollable list with buttons, each one of them is referring to a neuron, and a list of information for each neuron's stimuli.

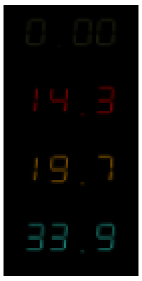
If the neuron receives an external stimulus then in the initial time a green colored number one (1) is shown. Conversely, a red colored zero (0) is shown.



Picture 4.12.1
Neuron's Stimuli Categories

The upper one consists of a horizontal scrollable list with the number of the time steps in hundreds. In order to watch the action potential of another hundred of time steps, the user can simply press the arrows in the left and the right side of the table. The single one, transfers him to the next or the previous one hundred, in opposition with the double arrow that transfer him by five hundreds.

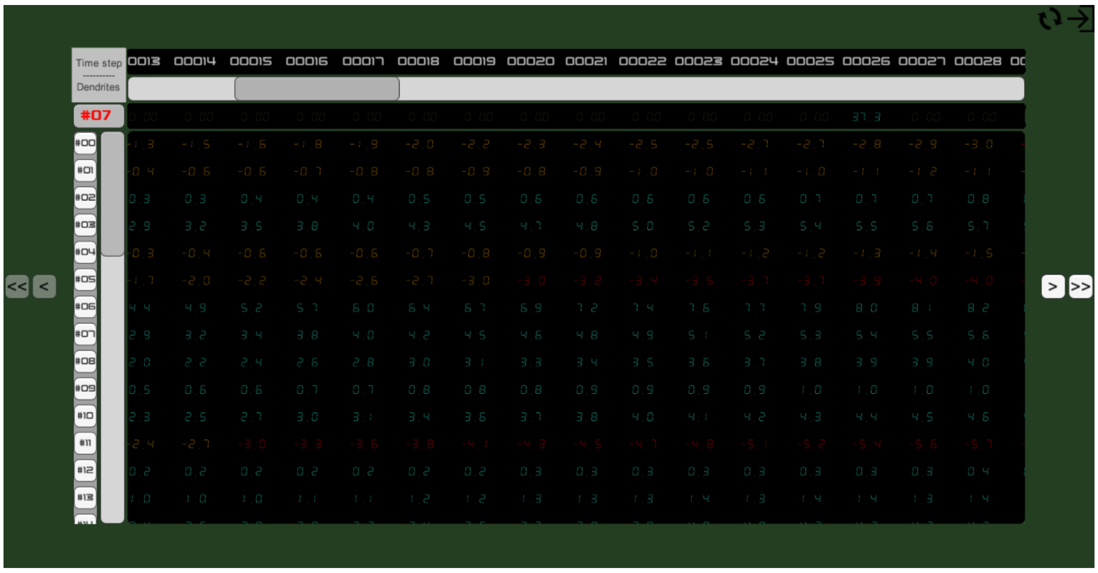
The main part shows the action potential of each neuron colored with different colors in order to be more discrete. There are four different colors. The brown stands for a zero (0) action potential. If the action potential is not equals to zero but it is much lower from the firing threshold then is colored with a red color. The yellow one, declares that the action potential is close enough to the firing threshold but it has not yet exceeded it. If it has exceeded it then it is colored with an ice-blue color.



Picture 4.12.2
Action Potential Categories

As it has already been mentioned, each one of the neurons is represented by a button in the vertical scrollable list. By pressing one of these buttons, the current table closes and a new one opens. This table contains the action potentials for this specific neuron's dendrites.

The structure of the table panel is similar to the previous one. The part of the neurons' stimuli is not included. Instead of this, there is a neuron button - colored based on its stimuli - and a horizontal scrollable list with the neuron's action potentials. This button is also used in order the user to return to the neurons' table.



Time step ---------- Dendrites	0013	00014	00015	00016	00017	00018	00019	00020	00021	00022	00023	00024	00025	00026	00027	00028
#07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	37.3	0.00	0.00
#00	-1.3	-1.5	-1.6	-1.8	-1.9	-2.0	-2.2	-2.3	-2.4	-2.5	-2.5	-2.7	-2.7	-2.8	-2.9	-3.0
#01	-0.4	-0.6	-0.6	-0.7	-0.8	-0.8	-0.9	-0.8	-0.9	-1.0	-1.0	-1.1	-1.0	-1.1	-1.2	-1.1
#02	0.3	0.3	0.4	0.4	0.4	0.5	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.8
#03	2.9	3.2	3.5	3.8	4.0	4.3	4.5	4.7	4.8	5.0	5.2	5.3	5.4	5.5	5.6	5.7
#04	-0.3	-0.4	-0.6	-0.6	-0.6	-0.7	-0.8	-0.9	-0.9	-1.0	-1.1	-1.2	-1.2	-1.3	-1.4	-1.5
#05	-1.7	-2.0	-2.2	-2.4	-2.6	-2.7	-3.0	-3.0	-3.2	-3.4	-3.5	-3.7	-3.7	-3.9	-4.0	-4.0
#06	4.4	4.9	5.2	5.7	6.0	6.4	6.7	6.9	7.2	7.4	7.6	7.7	7.9	8.0	8.1	8.2
#07	2.9	3.2	3.4	3.8	4.0	4.2	4.5	4.6	4.8	4.9	5.1	5.2	5.3	5.4	5.5	5.6
#08	2.0	2.2	2.4	2.6	2.8	3.0	3.1	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.9	4.0
#09	0.5	0.6	0.6	0.7	0.7	0.8	0.8	0.8	0.9	0.9	0.9	0.9	1.0	1.0	1.0	1.0
#10	2.3	2.5	2.7	3.0	3.1	3.4	3.6	3.7	3.8	4.0	4.1	4.2	4.3	4.4	4.5	4.6
#11	-2.4	-2.7	-3.0	-3.3	-3.6	-3.8	-4.1	-4.3	-4.5	-4.7	-4.8	-5.1	-5.2	-5.4	-5.6	-5.7
#12	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.4
#13	1.0	1.0	1.0	1.1	1.1	1.2	1.2	1.3	1.3	1.3	1.4	1.3	1.4	1.4	1.3	1.4

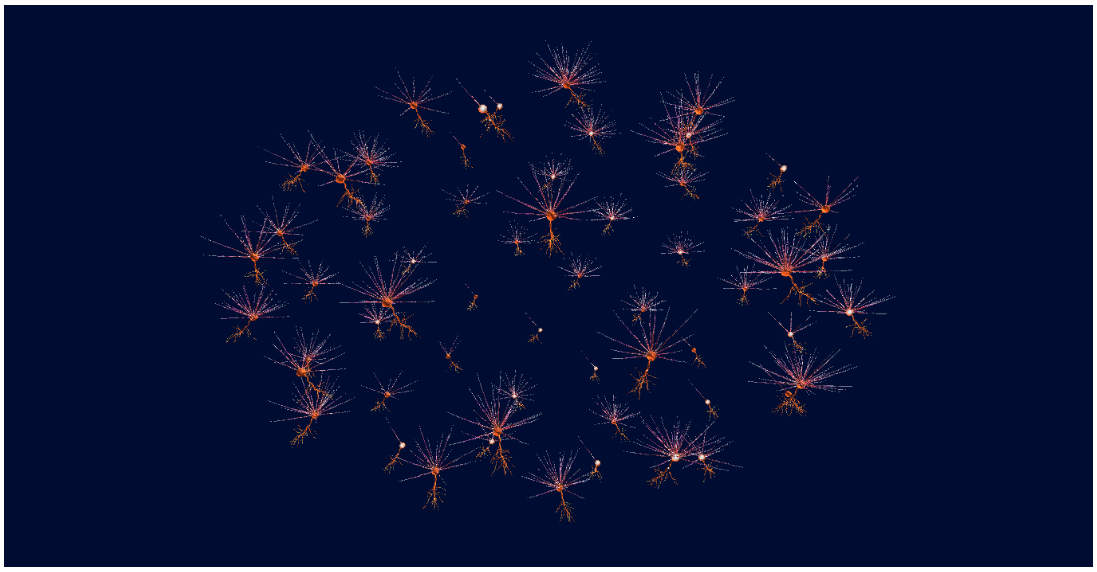
Picture 4.13
Table of a Neuron's Dendrites' Action Potential

Finally, an exit button in the upper-left corner of the page, leads the user back to the Welcome Page, whilst a change button transfers him to the second form of visualization (3D).

4.3 Three-Dimensional Visualization (UI)

The second form of visualization was based on the idea of the creation of a three-dimensional world, with three-dimensional neuron cells models in it.

The application enables the user to be transferred in space and time, by displaying specific parts of the neural network in specific time steps. Moreover, the user can be informed about the status of each neuron and each dendrite and has also the ability to watch plots with their action potential for every step.



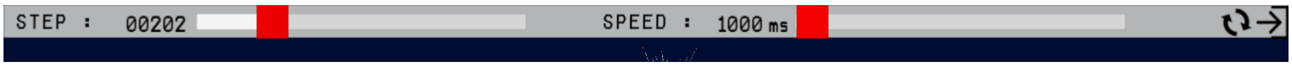
Picture 4.14
Three-Dimensional World

Initially, the three-dimensional world appears, along with two bars which disappear after two seconds.

Each one of the neuron cell models stands for one neuron out of the total number of neurons that the user has chosen to simulate. The same fact applies also to each one of the neuron cell's dendrites.

Neuron Cell models emit white light (light's intensity is associated with their action potential for each simulation step) and in case that the soma's firing threshold has been exceeded, the signal is transferred across their axon.
The excess of the dendrite's firing threshold is becoming visible through its model's color change. Its edges from purple are turned into white-ice.

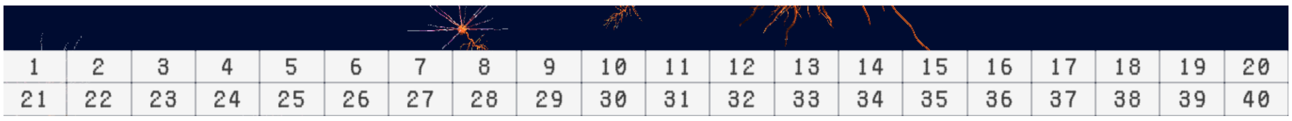
The use of the mouse enables the user to move along the space in three different ways:
a) Zoom-in / Zoom-out by scrolling the wheel of the mouse – moving in depth.
b) Rotation to every direction by holding right click pressed and moving the mouse - panning.
c) Moving to any direction without rotation or depth altering by holding the scroll wheel pressed and moving the mouse.



Picture 4.15
Hidden Task Bar

When the user places the mouse pointer on the top of the screen, the hidden bar above appears. This Bar - referred to as Task Bar - enables him to choose the desirable speed of the visualization by scrolling the right slider from one second (the default value) to four seconds. The left slider provides the ability of choosing (of) the current time of the visualization. Finally, an exit button in the right side of the bar leads the user back to the Welcome Page, whilst a change button transfers him to the first form of visualization (2D).

Moving the mouse pointer at the bottom of the screen another bar appears. This bar contains one button for every neuron and for this reason is referred to as Neurons Bar. By clicking on one of these buttons, a panel is displayed, the functions of which will be analyzed later on.



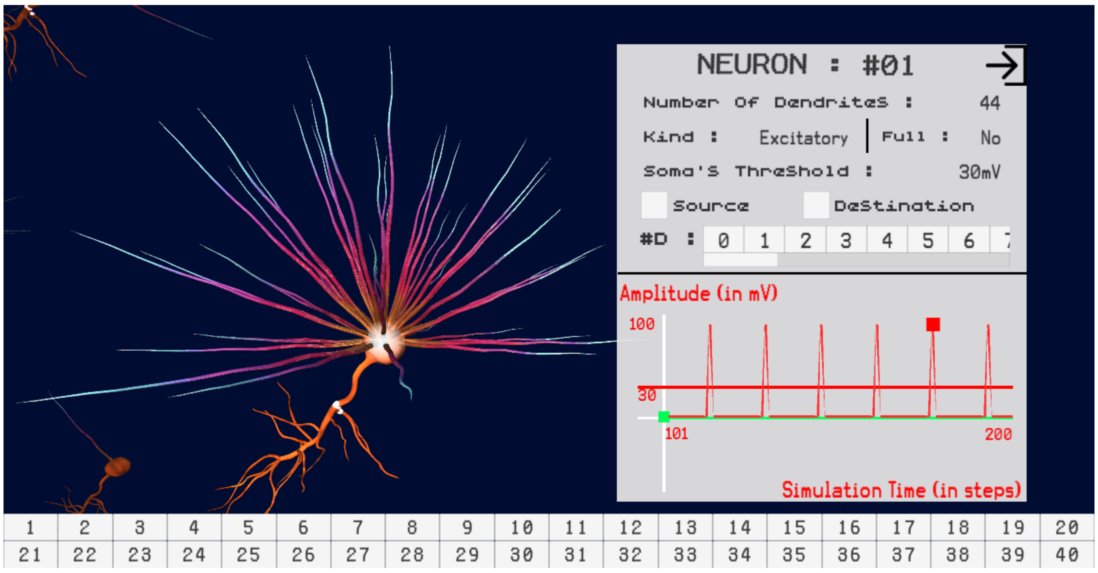
Picture 4.16
Hidden Neurons Bar

Clicking one of the buttons in Neurons Bar or picking with the mouse one of the neuron cell models, the screen is changing form.

In the left half the user can see the specific neuron cell model with all its dendrites. In the right half a panel appears. This panel contains every information on the neuron, as well as a plot of soma’s action potential.

The commands of moving in 3D space do not apply in this step. The user can use the

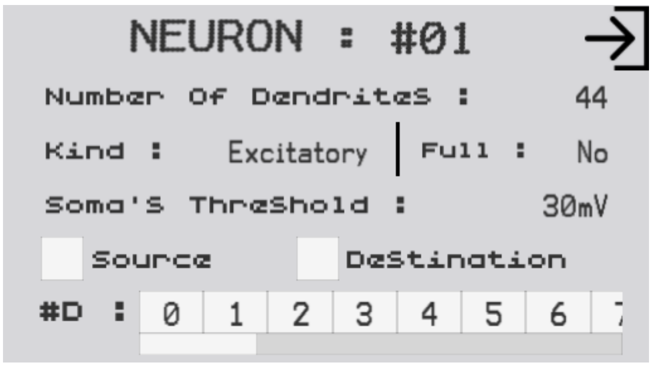
mouse in order to watch the model from every side by rotating the screen in y axis (vertical) and in z axis (depth):
a) Y axis rotation by 5 degrees pressing right click.
b) Z axis rotation by 5 degrees pressing the scroll wheel.



Picture 4.17
Neuron Panel

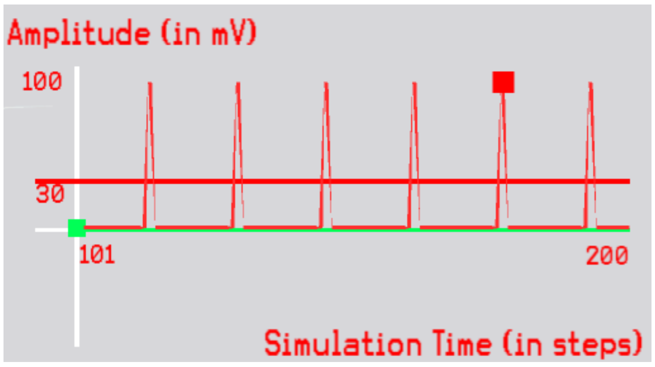
The left-half panel can be divided in two parts. The upper one comprise information about the neuron cell. Specifically:

- The number of dendrites this neuron contains.
- The neuron's kind (Inhibitory / Excitatory).
- The fullness of its dendrites' synapses.
- The soma's threshold.
- Two checkboxes labeled Source / Destination. The first one provides the user with the information about its presynaptic neurons. The second one is about the postsynaptic neurons (red/green-colored buttons respectively, whilst yellow color declares both).
- A button - list of its dendrites.



Picture 4.17.1
Neuron's Information

The lower-half panel, contains a plot about the soma's action potential in the current hundred of simulation steps. A red line stands for the soma's firing threshold whilst the green is for the background, the extra potential than neuron receives in every step. There are also two points, a red one that stands for the current step's action potential and a green one for the neuron's stimulus.

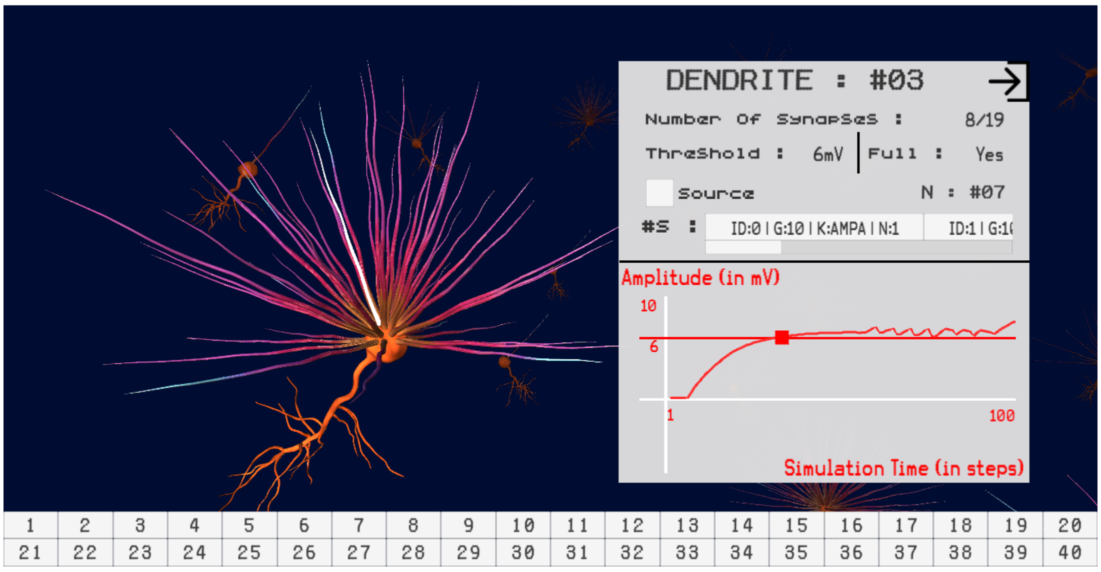


Picture 4.17.2
Neuron's Plot

An Exit button in the upper-right corner leads the user back to three-dimensional space.

Clicking one of the buttons in Dendrites button-list or picking with the mouse clicking on one of the dendrite models, the panel in the right half is changing form. The chosen dendrite is highlighted with white color.

The commands of screen rotation are applied to this step too.



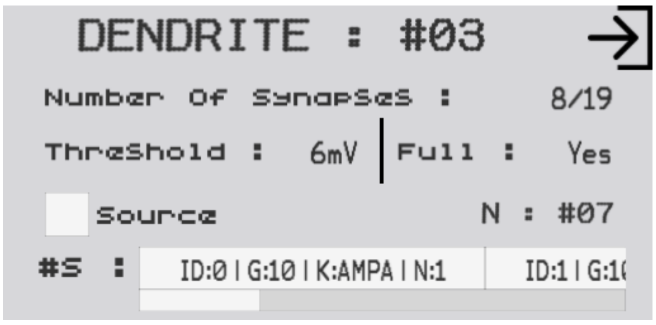
Picture 4.18
Dendrite Panel

The upper-half of the panel contains information about the specific dendrite:

- The number of synapses this dendrite contains and which ones are connected to a neuron.
- The dendrite's threshold.
- The fullness of dendrite's synapses.
- One checkbox labeled Source to provide the user with the information on the neurons

that this dendrite is connected with (red colored button).

- A list of its full synapses (including information about their kind, grouping and neuron that each one of them is attached to).



DENDRITE : #03

Number Of Synapses : 8/19

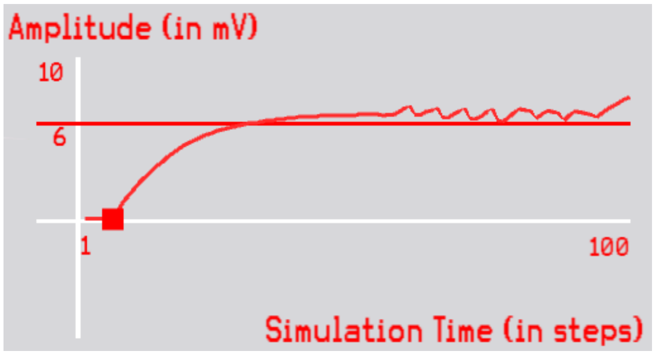
Threshold : 6mV | Full : Yes

Source N : #07

#S : ID:0 | G:10 | K:AMPA | N:1 ID:1 | G:1

Picture 4.18.1
Dendrite's Information

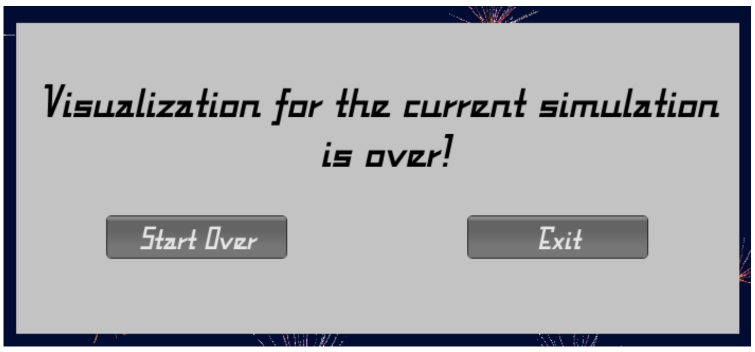
The lower-half panel, contains a plot about the dendrite's action potential in the current hundred of simulation steps. A red line stands for the dendrite's firing threshold and a red point stands for the current step's action potential.



Picture 4.18.2
Dendrite's Plot

An Exit button in the upper-right corner leads the user back to the Neuron Panel.

When the visualization is completed a pop-up window gives the user the ability to choose between repeating the visualization or returning to the Welcome Page.



Visualization for the current simulation is over!

Picture 4.19
Repeat or Exit Pop-up

Chapter 5: Implementation

This chapter constitutes a detailed description of the implementation of each part of the application. As well as the previous chapter, the current chapter will also be analyzed in three parts. The first one, in which the input of the data takes place and the next two parts that are related to the management of the output data. For each one of these parts, a different Unity Scene has been designed.

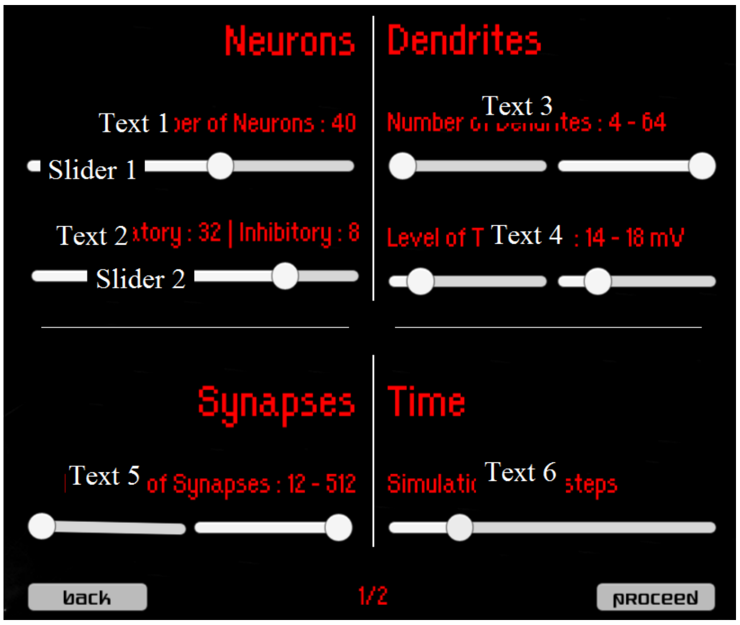
Because some terms are difficult and some may have trouble comprehending them or even find them confusing, it is clarified in advance that whatever includes the word "Unity" in any way and is highlighted, constitutes a **Unity** component.

5.1 Welcome Scene - Scene 0

The first **Unity Scene** is a two dimensional one. The main **Unity GameObjects** from which it is composed, are a **Main Camera** and a **Canvas**. The **Canvas** contains an amount of **Unity UI** components whose functionality is regulated by C# scripts that are attached to them. The **Main Camera** is a static one, so there is no need for a C# Script in order to control its functions.

Initially, a .jpeg file [23] has been modified in order to be used as a background image. For each one of the seven steps that were described in the first subchapter of the previous chapter, a **Unity Panel (UI)** has been created.

A **Unity Animator** combined with a C# script titled <u>Menu_Manager</u> (is attached to **Canvas**) are responsible for the switching process between the **Panels**. The **Animator** has been designed in order to contribute to the smooth transitions among the **Panels**, while the Menu_Manager manages in which **Panels** the **Animator** will be applied.

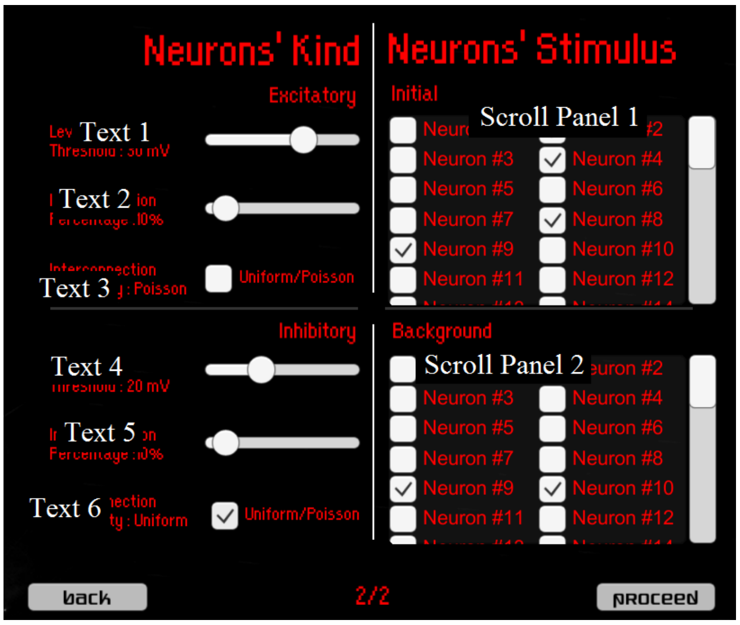


Picture 5.1
Input of Parameters Panel (1)
-Step three -

The first two **Panels** (Welcome Page, Form of Action) simply consist of **Unity UI Texts** and **Buttons**. Due to the fact that **Unity Engine** greatly facilitates the application developer, from now on only the parts of implementation that require further development will be described.

The Input of Parameters (1) Panel additionally contains **Unity Sliders (UI)**. Furthermore, every action in this step is adjusted by three C# scripts.

- The One_Slider_to_Text .cs file is used in order to display a specific **Slider's** value as a text (is attached to **Texts** 1 and 6).
- The **Slider** 1 adjusts the desirable value for the number of the neurons. The **Slider** 2 regulates the percentage of each one of their kind. The Neurons_Slider .cs file presents as a Text the quantity of the Excitatory and the Inhibitory neurons(is attached to **Text** 2).
- The Two_Sliders_to_Text .cs file is incorporated in **Texts** 3, 4 and 5. This .cs file is responsible for displaying the minimum and the maximum value of a parameter (left and right **Slider** respectively), and it also sees to the minimum value not to be over the maximum and vice versa.



Picture 5.2
Input of Parameters Panel (2)
-Step four -

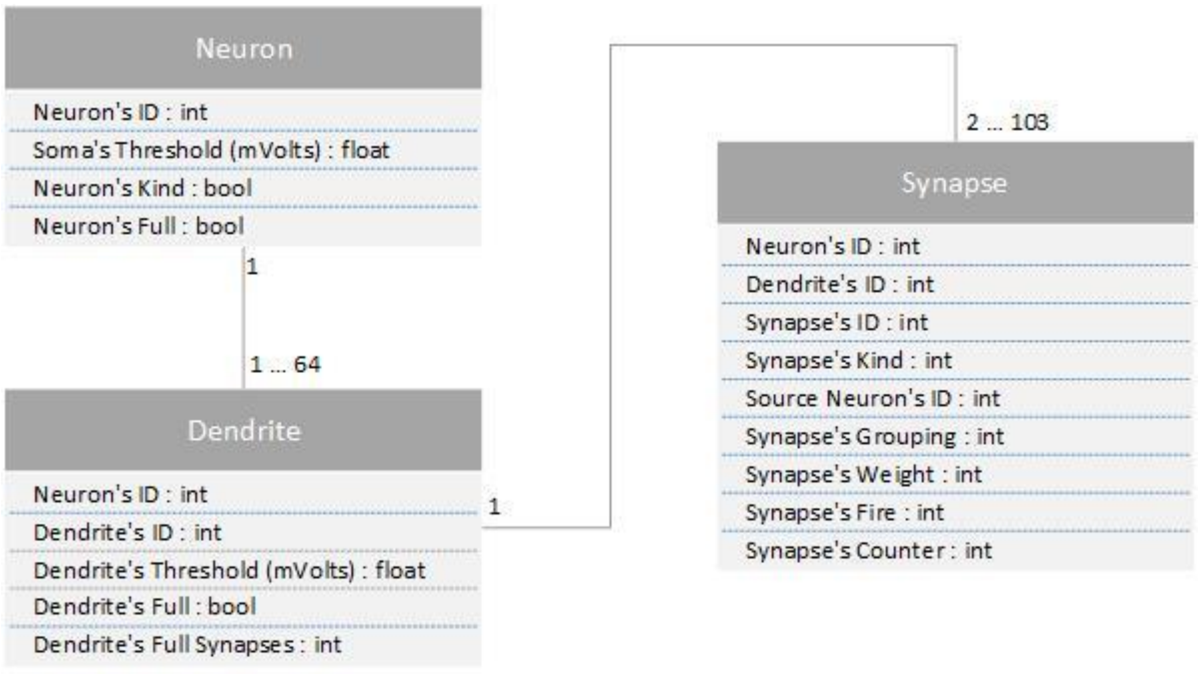
The new components that were used in the design of the second Input of Parameters Panel are the **Unity Toggle (UI)** and the vertical Scroll Panel, which requires the use of a **Unity Scrollbar (UI)**.

Apart from the aforementioned One_Slider_to_Text (attached to **Texts** 1, 2, 4 and 5) this panel's functionality requires two others C# scripts.

- One_Toggle_to_Text .cs file is incorporated to **Texts** 3 and 6 and it is responsible for displaying **Toggle's** value as a text.
- Design and management of the Scroll Panels are adjusted by using the Neurons_Stimulus .cs file which is incorporated to these. For the **Toggles** contained in them, instances of pre-created **Unity GameObjects** have been used.

The **Unity Panel** of the Loading Page, is simply composed of a **Text** and a **Slider**. However, the most crucial C# script for the continuation of the program is attached to this panel. This script, named File_Creation, shoulders the entire process from the moment the user enters all of the simulation parameters until the moment that the user will select the visualization form of the output data .

Initially, for each one of the units of the simulation (neurons, dendrites and synapses) a C# Struct has been created. These structures, comprise every necessary for the simulation parameters, mostly based on the values of the **Sliders** and the **Toggles** from the two previous steps.



Picture 5.3
Data Structures

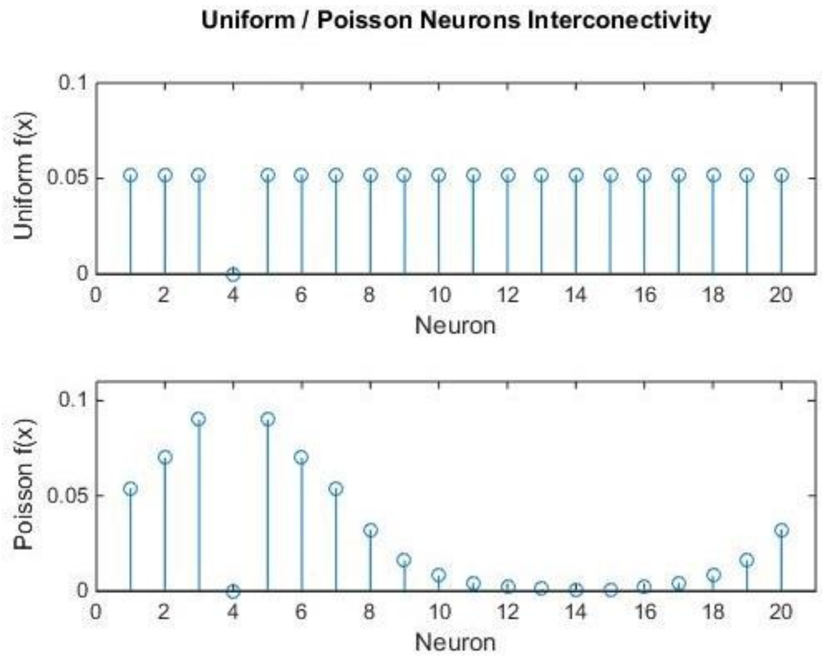
The ID Number of every C# Struct is a serial number (starts with one (1) for neurons and with zero (0) for dendrites and synapses). In case a parameter has two or more equally likely possible values or if the value of a parameter has to be between a minimum and a maximum value (like the number of neuron's dendrites or the number of dendrite's synapses), then the Uniform Distribution is used. Uniform Distribution is used for the following parameters:

– Neuron's Kind: It refers to whether a neuron is inhibitory or excitatory. Its kind also determines the Soma's Threshold.
– Dendrite's Threshold: The firing threshold for a Neuron's Dendrite. Even if two dendrites belong to the same neuron cell, they could have dissimilar firing threshold.
– Synapse's Kind: A synapse could be AMPA (40%), NMDA (40%), GABBa (10%), GABBb (10%). Each of one them has a different kind of synaptic curves.
– Synapse's Grouping: It creates a group of synapses that can contain either five or ten synapses. Grouping is responsible for the actual number of synapses that a dendrite can contain. Even though initially, the minimum limit can be 12 synapses and the maximum 512 synapses, grouping process creates new but almost impossible limits: the minimum of 2 actual synapses (2 groups of ten) and the maximum of 103 actual synapses (103 groups of five).

- Synapse's Weight: It stands for the amount of electrical activity which will be transmitted by the synapse (get values from one (1) to four (4)).

C# Struct Synapse's variables, fire and counter, are set to zero (0). They are only used by the simulation algorithm. Neuron's Full, Dendrite's Full, Dendrite's Full Synapses & Source Neuron's ID are defined after the interconnection process between the neurons has taken place.

As soon as the above-mentioned variables are filled, the process of interconnection between the neurons follows. Each neuron's kind determines the way that it will be connected to the other neurons, according to two factors. The first one is the interconnection percentage, which is relevant with the amount of the post-synaptic neurons that this pre-synaptic neuron is attached with. The second one is the interconnection probability. The only difference between Poisson and Uniform Distribution, is the fact that by using the first, the pre-synaptic neurons are more possible to be connected with a neighboring post-synaptic neuron.



Picture 5.4
Distributions: Poisson / Uniform
(Neuron 4 in a sum of 20 Neurons)

The part of the interconnection also sets the following variables:

- Neuron's Full: If every dendrite's synapses of the specific neuron is occupied, then the neuron is considered to be full.
- Dendrite's Full: If every synapse of the specific dendrite is connected to another neuron, then the dendrite is considered to be full.
- Dendrite's Full Synapses: This variable declares the number of a dendrite's synapses that are occupied by a pre-synaptic neuron.
- Source Neuron's ID: The Number ID of the pre-synaptic neuron which is attached to this specific synapse.

At this point, every variable from every C# Struct has been filled out. The next step is the creation of every file that is going to be needed. The files are divided in three categories:

- files that will be uploaded to the server
- files that will be saved in order to be used for the visualization
- files that help the process

Simultaneously, into the local disk's path "USERNAME\Documents\Visualization Neural Network\" two folders have been created in order to host these files:

- Extra Simulation Files
- Simulations

Files that will be uploaded to the server and files that will be used for the visualization part of the application, are located into a folder that is named by the date and the time (in the form yy_MM_dd_HH_mm_ss) that the simulation took place. This specific folder is located into the "Extra Simulation Files" folder.
Four different files are created in order to be uploaded:

- thresholds.txt (server file): All of the neuron cells soma's and dendrite's firing thresholds values are contained in this file.
- convey_init.txt (server file): It contains information about every synapse.
- convey_update.txt (server file): This file has information about the interconnection between the neurons.
- input_parameters.txt (server file): Stimuli and Background active neurons are declared in this file together with the simulation duration.

The first three files are encoded according to the form that is detailed below. The five most significant bits stand for a tag number – that differs between the files. Bits that are not omitted are unused. The 64bit binary number will be written in the file after its conversion into decimal.

Neuron's ID	Dendrite's ID	Neuron's Threshold	Dendrite's Threshold
51 downto 45	44 downto 39	31 downto 16	15 downto 0

(a) thresholds.txt – with tag number "00000"

Neuron's ID	Dendrite's ID	Synapse's ID	Synapse's Counter
51 downto 45	44 downto 39	38 downto 32	25 downto 16
Synapse's Weight	Synapse's Step	Synapse's Kind	Synapse's Fire
15 downto 12	11 downto 4	3 downto 1	0

(b) convey_init.txt – with tag number "00001"

Source Neuron's ID	Neuron's ID	Dendrite's ID	Synapse's ID
58 downto 52	51 downto 45	44 downto 39	38 downto 32

(c) convey_update.txt – with tag number "00011"

Picture 5.5
Encoding Data

The fourth file contains three different decimal numbers splitted by a space. The first one declares the number of simulation steps. The other two numbers are decimal conversions of two binary numbers (the number of (the) digits of each one equals to the number of the neurons), and its digits values depend on the existence or not of stimuli and background respectively.

Except from the files that will be uploaded to the server, there are files that are created in order to be used in the parts of visualization:

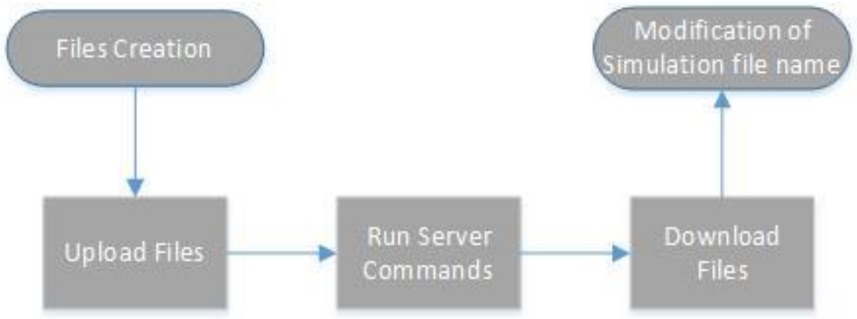
- fileNeurons.txt (visualization file): Every information for every neuron could be found in this file.
- fileDendrites.txt (visualization file): This file contains every variable that is included in every structure Dendrite.
- fileSynapses.txt (visualization file): In this file is written every information about all the synapses.
- fileSummary.txt (visualization file): Contains the number of neurons and the number of simulation's steps values.

Finally, two more files are created. The first one (referred to as Simulation file) is initially named by the date and the time (in the form yy_MM_dd_HH_mm_ss) and is saved into the folder "Simulations". The name of the folder that contains all of the simulation's files (server and visualization) is written in the Simulation file.

The second file is called help_file.txt. It is saved in the same folder as the .exe file. The name of the folder that contains all of the simulation's files (server and visualization) is also written in this, but only temporarily, as it will be subsequently analyzed.

After every useful file is created, the application uses Renci.SshNet.dll [24] in order to communicate with the server in which the simulation's implementation is located. The interaction between the application and the server is conducted in three steps:

- files uploading : The four server files that were created are being uploaded on to the server
- commands running : Specific commands are running and the server provides results
- files downloading : Two files are being downloaded into the folder that contains the server and visualization files for the current simulation. The first one (results_cell_data.txt) contains the action potential's value of every soma for every simulation step that it is not equal to zero. The second one (results_dendrite_data.txt) includes that same information about every dendrite



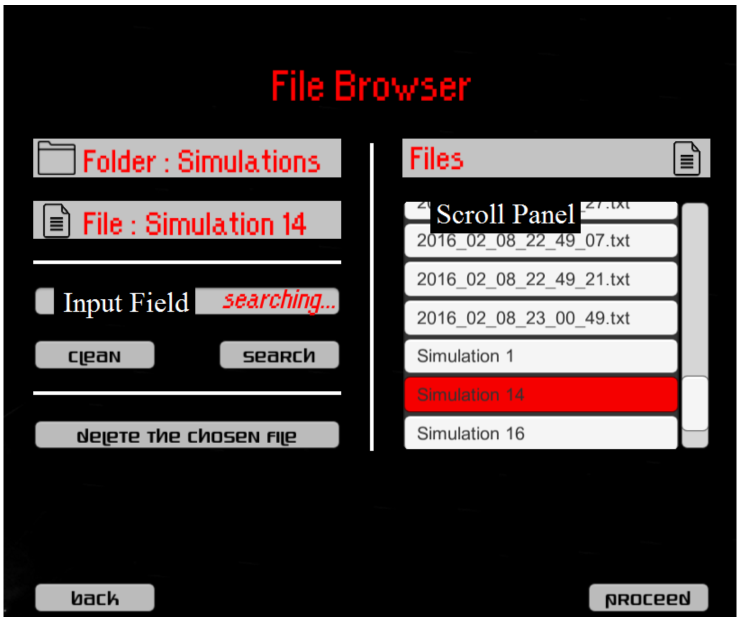
Picture 5.6
Server Steps Diagram

In this .cs file an OnGui() function is also included. Its functionality is to display the following messages for every step that is completed in order to inform the user about the process:

- "Files Uploaded..."
- "Simulation Ended..."
- "FPGA Execution Time : () seconds"
- "Files Downloaded..."

For the parts of files creation and server interaction a thread has been used to improve the speed of the execution.

The second .cs file that is attached to the current **Unity Panel** Loading Page, takes command of displaying a loading bar to the user. When the files are downloaded from the server, the loading bar will stop loading. By using an OnGui() function, a pop up Window enables the user to modify the Simulation file's name, which originally is named by the date and the time of creation. After the name editing the file will continue to be included into the folder "Simulations".



Picture 5.7
File Browser Panel
-Step six -

The penultimate Panel that was created, constitutes a browser for all the Simulation files (located into the folder "Simulations"). Its functions are regulated by File_Browser_Handler .cs file.

The only new component that is contained in this panel, is a **Unity Input Field (UI)**. The Scroll Panel in the left-half, includes instances of pre-created **Unity GameObjects**, specifically **Unity Buttons (UI)**. Each one of these buttons represents one of the Simulation files.

The list of the Simulations' files was created by using the System.IO.Directory.GetFiles (pattern) – the factor "pattern" is the string variable *.txt - function. The factor "pattern" of the function is being modified according to the content of the **Input Field** – in this case it will be the string variable *input_field*.txt.

By using an OnGui() function, a pop-up window (requires) inquiries about confirmation in order a Simulation file to be deleted, given that a file is chosen. In this case the folder in the Extra Simulation Files that is related to this simulation, will be also deleted.

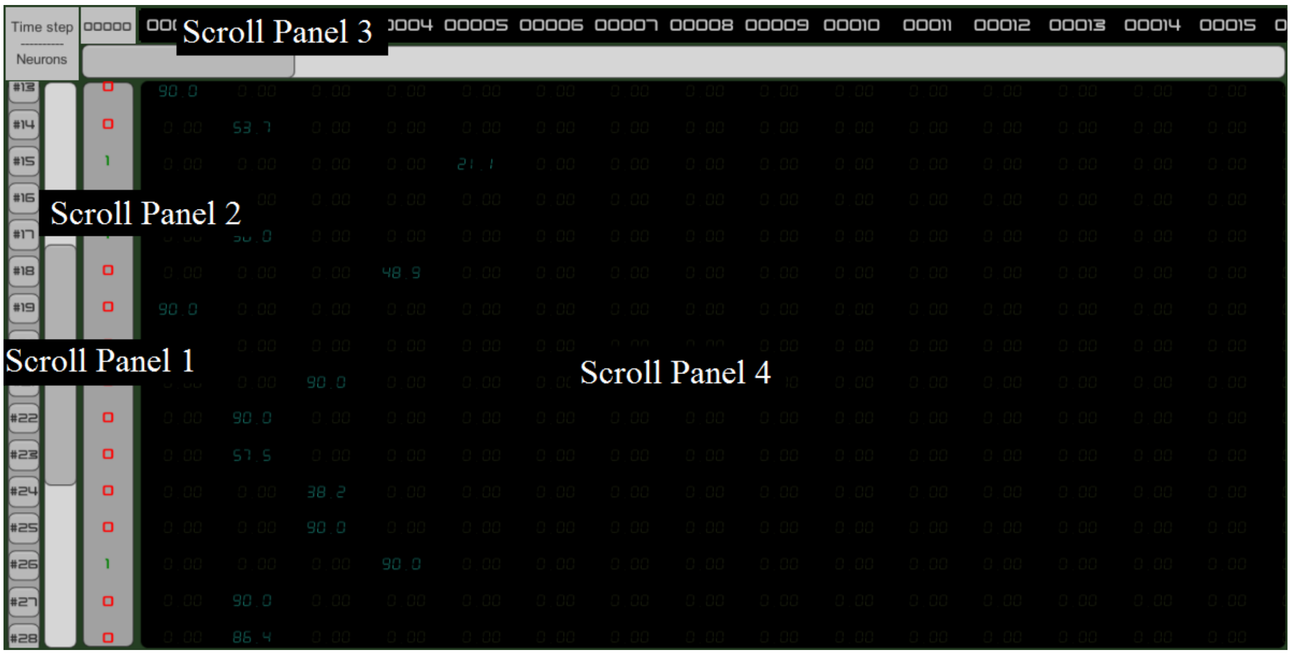
The choosing of a Simulation file modifies the help_file.txt. The name of the folder that contains this specific simulation's files will be written into the help_file.txt.

Finally, it should be pointed out that as Menu_Manager is responsible for the switching between the **Panels**, there is also a Scenes_Manager .cs file that contributes to the transitions between the three **Unity Scenes**.

5.2 Two-Dimensional Visualization - Scene 1

The second Unity Scene is also a two dimensional one. The main **Unity GameObjects** from which it is composed of, are a **Main Camera** and a **Canvas (UI)**. The **Canvas** contains an amount of **Unity UI** components whose functionality is regulated by C# scripts that are attached to them. The **Main Camera** of this **Scene** is also a static one.

Initially, in order to proceed with the displaying of the desirable data, the recovery of the information which is written in the related files - as it was described previously - is required. The Read_Files .cs file contains the code which is responsible for doing this.



Picture 5.8
Table of Neurons' Action Potential – Step one -

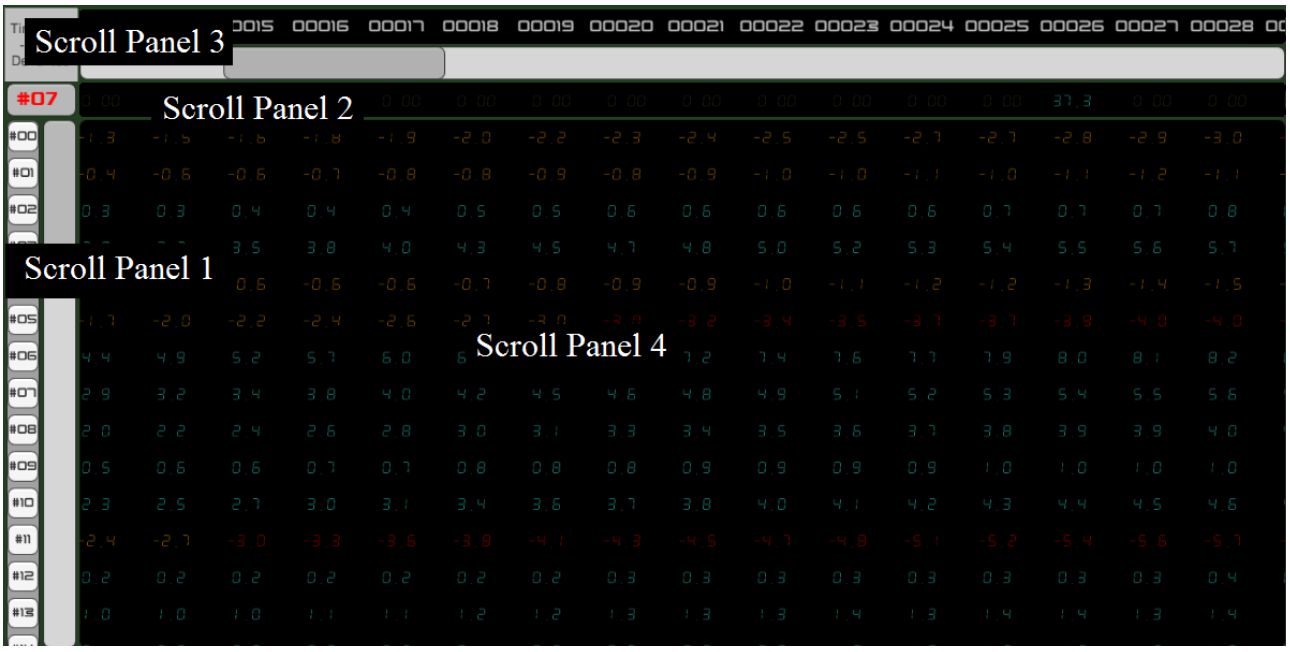
Firstly, the help_file.txt, which is located in the same folder as the .exe file of the application, is read. In this file the name of the folder (into the Extra Simulation Files) in which the files related to the simulation - that the user wants to display its results-are located, is contained. The four files with the information about neurons, dendrites, synapses, simulation time, as well as the two files with the output data that were downloaded from the server are read, and their content is written into arrays. For this part, a thread has been used to improve the speed of the execution.

For each one of the tables that were described in the second subchapter of the previous chapter, a **Unity Panel** has been created. The **Unity Animator**, combined with the C# script Menu_Manager, is used for the switching between the **Panels** (Table of Neurons' Action Potential and Table of Neuron's Dendrites' Action Potential). The **Animator** has been designed in order to contribute to the smooth transitions between the **Panels**, while the Menu_Manager manages in which **Panels** the **Animator** will be applied.

The first of two **Panels** – which is related with the neurons' action potential - is divided into four Scroll Panels, handled by a vertical and a horizontal **Unity Scrollbars (UI)**.

- Scroll Panel 1 is a vertical one. It contains instances of pre-created **Unity Buttons (UI)** and each one of them represents a neuron.
- The second Scroll Panel is also a vertical one. It consists of instances of **Unity Texts(UI)**, in which the number zero (0) or (1) is written indicating the existence or not of stimuli.
- Scroll Panel 3 is horizontal and it displays the simulation steps, using instances of **Unity Texts (UI)**.
- The fourth Scroll Panel also includes instances of **Unity Texts (UI)**, but it can be scrolled both horizontally and vertically. In each one of its Texts, a float number that is related to a neuron's soma action potential is written.

For each one of these panels, a C# script has also been created. These .cs files are responsible for displaying each panel’s instances and their contents.



	[illegible]	[illegible]	0015	00016	00017	00018	00019	00020	00021	00022	00023	00024	00025	00026	00027	00028
#07	0.00	[illegible]	[illegible]	[illegible]	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	37.3	0.00	0.00
#00	-1.3	-1.5	-1.6	-1.8	-1.9	-2.0	-2.2	-2.3	-2.4	-2.5	-2.5	-2.7	-2.7	-2.8	-2.9	-3.0
#01	-0.4	-0.6	-0.6	-0.7	-0.8	-0.8	-0.9	-0.8	-0.9	-1.0	-1.0	-1.1	-1.0	-1.1	-1.2	-1.1
#02	0.3	0.3	0.4	0.4	0.4	0.5	0.5	0.6	0.6	0.6	0.6	0.6	0.7	0.7	0.7	0.8
[illegible]	[illegible]	[illegible]	3.5	3.8	4.0	4.3	4.5	4.7	4.8	5.0	5.2	5.3	5.4	5.5	5.6	5.7
[illegible]	[illegible]	[illegible]	0.6	-0.6	-0.6	-0.7	-0.8	-0.9	-0.9	-1.0	-1.1	-1.2	-1.2	-1.3	-1.4	-1.5
#05	-1.7	-2.0	-2.2	-2.4	-2.6	-2.7	-3.0	-3.0	-3.2	-3.4	-3.5	-3.7	-3.7	-3.9	-4.0	-4.0
#06	4.4	4.9	5.2	5.7	6.0	6	[illegible]	[illegible]	7.2	7.4	7.6	7.7	7.9	8.0	8.1	8.2
#07	2.9	3.2	3.4	3.8	4.0	4.2	4.5	4.6	4.8	4.9	5.1	5.2	5.3	5.4	5.5	5.6
#08	2.0	2.2	2.4	2.6	2.8	3.0	3.1	3.3	3.4	3.5	3.6	3.7	3.8	3.9	3.9	4.0
#09	0.5	0.6	0.6	0.7	0.7	0.8	0.8	0.8	0.9	0.9	0.9	0.9	1.0	1.0	1.0	1.0
#10	2.3	2.5	2.7	3.0	3.1	3.4	3.6	3.7	3.8	4.0	4.1	4.2	4.3	4.4	4.5	4.6
#11	-2.4	-2.7	-3.0	-3.3	-3.6	-3.8	-4.1	-4.3	-4.5	-4.7	-4.8	-5.1	-5.2	-5.4	-5.6	-5.7
#12	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.4
#13	1.0	1.0	1.0	1.1	1.1	1.2	1.2	1.3	1.3	1.3	1.4	1.3	1.4	1.4	1.3	1.4

Picture 5.9
Table of Neuron's Dendrites' Action Potential – Step two -

The second of two **Panels** – which is related to a specific neuron's dendrites' action potential - also consists of four Scroll Panels – handled by a vertical and a horizontal **Unity Scrollbar (UI)**.

The third and the fourth Scroll Panel are the same as in the previous **Panel**. Conversely, the other two are a little bit different:

- The first one contains instances of pre-created **Unity Texts (UI)** instead of **Buttons**, each one of them representing a neuron's dendrite.
- The second Scroll Panel is a horizontal one. In each one of its **Texts**, a float number that is related to a specific neuron's soma action potential is written.

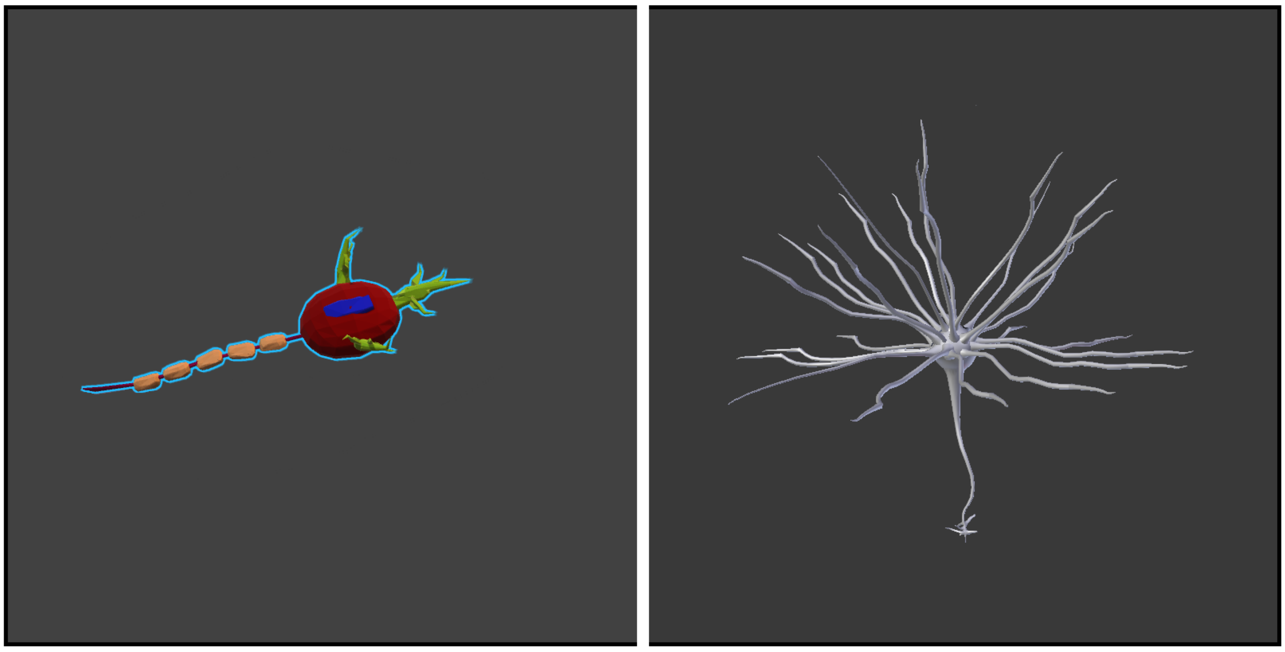
The second **Panel** also contains a **Button (UI)**, the management of which – as well as the other functions of the specific **Unity Scene** – is regulated by Scene_Handler .cs file.

The current **Scene**, as the previous ones, also contains a Scenes_Manager .cs file that contributes to the transitions among the three **Unity Scenes**.

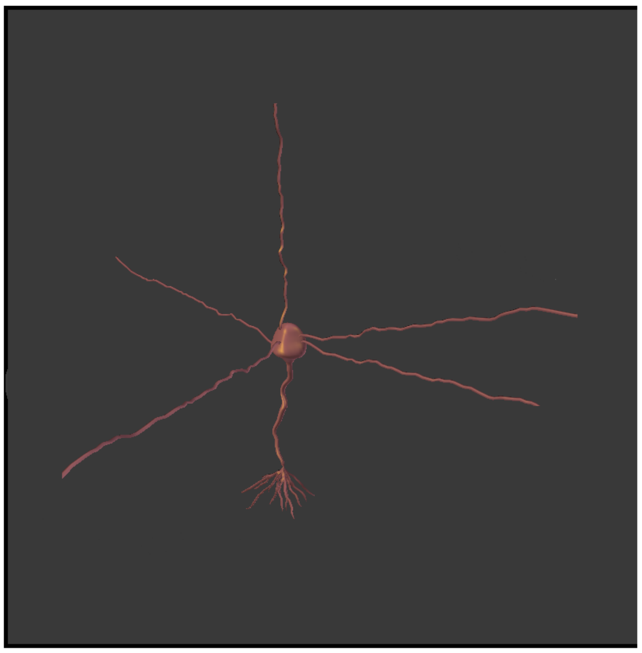
5.3 Three-Dimensional Visualization - Scene 2

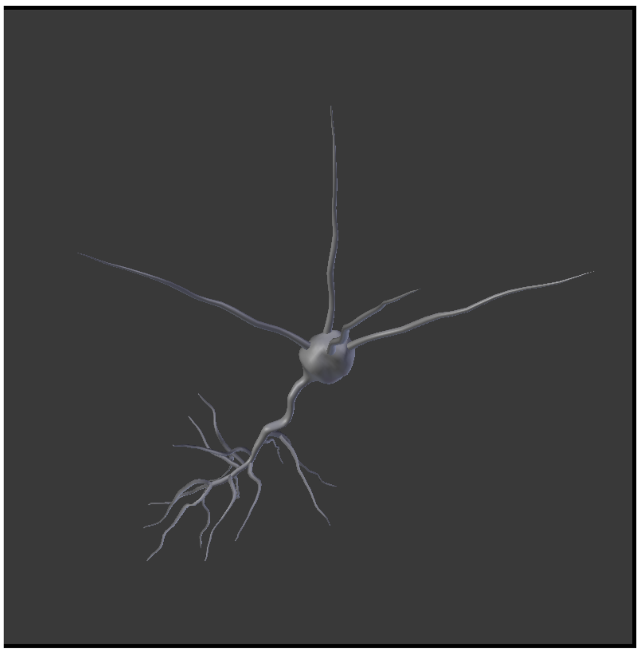
The design of the appropriate three-dimensional models that were used for the particular part of visualization, constitutes a separate section of the implementation of the application. Altogether, four different three-dimensional models were designed, until the last one to be selected as more realistic and more aesthetic.

It is noteworthy that in the last two three-dimensional Neuron-Cell models, the Soma's model has been designed separately from that of the Dendrite's/the Dendrite's one. In this way the cell model can be presented with the exact number of dendrites, which was determined before the part of the simulation.


(a) Using Autodesk 3ds Max 2015 *(b) Using Blender 2.7x*

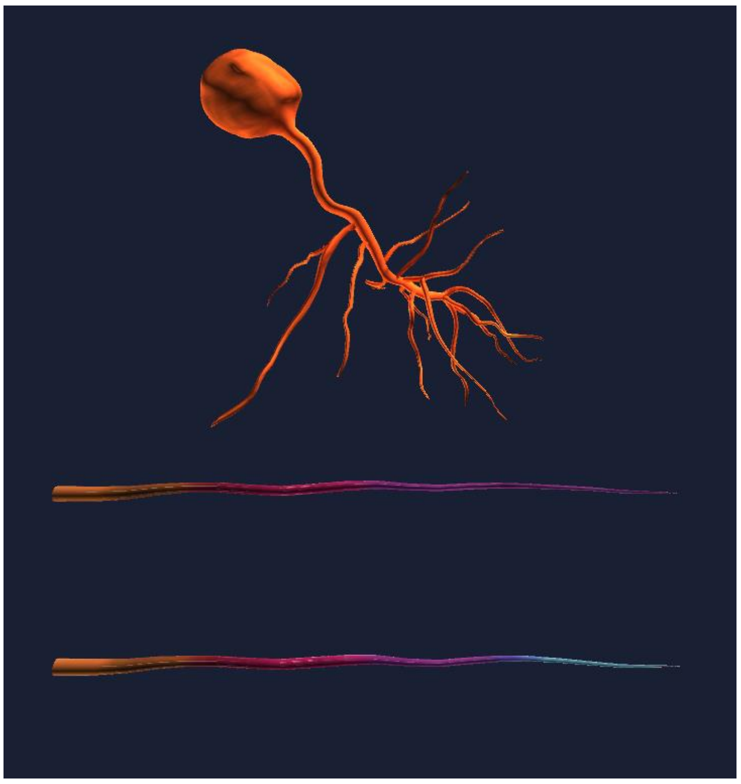
Picture 5.10
Three-Dimensional Neuron-Cell Models
- With dendrites attached to them


(a) Using Blender 2.7x - Soma & Dendrite Model


(b) Using Blender 2.7x - Soma & Dendrite Model

Picture 5.11
Three-Dimensional Neuron-Cell Models
– Dendrites as a separate model

Two different 3D computer graphics software were used for the design of the above models, as well as a 3D sculpting and painting tool for the editing of the final model.


Picture 5.12
Final Models after editing (with Unity Material)
– using Autodesk Mudbox 2015

The three-dimensional version of visualization of the output data constitutes, as it was referred in the beginning of the chapter, a separate **Unity Scene**. The basic components that it is composed of, except for the **Unity Canvas** and the **Main Camera** – that are contained in both of the previous Scenes – are:

- Three **Unity Lights**
- Pre-created **GameObjects** that are used for the **Canvas' Panels**
- One pre-created **GameObject**, in which the neuron-cell model is included.
- Four pre-created **GameObjects**, in which the dendrite model is contained.
  Each one of them will be separately analyzed subsequently.

As already mentioned in the begging of the previous subchapter, the C# Script Read_Files plays the primary role in recovering the data from the files in which they are written. The way was described in the previous **Unity Scene**.

As soon as the basic elements – like the number of the neurons or the number of simulation steps - are (known) notified, the design is ready to begin. Create_Neurons .cs file undertakes – based on the data - the creation and the representation in the screen of the three-dimensional models. In this case, two algorithms are used:

- The first one is related to the neuron-cell model and its purpose is to deposit each one of the models into the periphery of a sphere [25] into the three-dimensional world, so that each one of them to be equidistant from its neighboring. Neighboring is based on the serial Neuron's ID. For example, for twenty neuron-cells, Neuron 4 is spatially closer to the Neurons 7 and 20, than to Neurons 12 and 15.

```
Distribute_Points (int points)
{
        $radius = 120$
        $i = \pi * (3 - \sqrt{5})$
        $j = {}^{2}/_{points}$

        $for\ k = 0, k < points, k++$
        {
                $y = k * j - 1 + ({}^{j}/_{2})$
                $r = \sqrt{1 - y^2}$
                $\varphi = k * i$
                $NeuronPosition(x, y, z) = r * \cos\varphi, y, r * \sin\varphi$

        }

        Check_Clusters (points)
}

Check_Clusters (int points)
{
        $for\ k = 0, k < points, k++$
        {
                Check NeuronPosition(x, y, z) for Neuron(k)
                Put Neuron(k) in one of the eight Clusters
        }
}
```

Picture 5.13
Algorithm of Neuron's Positioning

- The second algorithm regulates the positioning of a neuron's dendrites according to Dendrite's ID. The higher into the cell a dendrite is put, the lesser its Dendrite's ID is.

```
DendritesRotationXYZ = {# 64 points (x,y,z), for every one of the 64 possible dendrites}

DendritesOrder = {# 64 numbers represent the order for each one}

Check_DendritesOrder (int number_of_Dendrites)
{
        for k = 0, k < number_of_Dendrites, k + +
        {
              newDendritesArray [i]  =  DendritesOrder[i]
        }

       Sort (newDendritesArray)
}
```

Picture 5.14
Algorithm of Dendrite's Positioning

The second most important C# Script is named Time and is related to everything that has to do with the time – speed of visualization, simulation step. Specifically, it uses **UnityEngine.Time.deltaTime** in order to increase the step by one, whenever time exceeds the visualization's speed. This .cs file is also responsible for every change of the Step and the Speed Sliders.

When the visualization has completed, by using an OnGui() function, a pop-up window enables the user to choose between visualization's repeating or returning to the initial **Scene**, that was described on the first subchapter of the current chapter.

Even if this form of visualization is based on the representation of a three-dimensional world, a two-dimensional **Unity Canvas** is used in order to analytically display every necessary data. In this **Canvas**, four separated **Unity Panels** are contained. Responsible for the presentation of its contents is a C# Script called Extra_Windows_Handler.

The first upper horizontal panel consists of two **Unity Sliders (UI)** and two **Unity Buttons (UI)**. The two **Sliders** are regulating the user's intended step of as well as the visualization speed that is the period that the visualization of a simulation step will last. As already mentioned, the Time .cs file is responsible for the **Sliders**. The two **Buttons** are used for the switching between the **Scenes**.

The lower horizontal one, consists of a number- equal to the number of neurons – of pre-created **Unity Buttons (UI)**. Its functionality has been analyzed in the third subchapter of chapter 4. Neurons_Button .cs file is attached to this **Panel** and it adjusts the positioning of the **Buttons** into it.

The remaining two of the Panels (Neuron and Dendrite Panels) are those that consist of the chosen neuron and the chosen dendrite's information and chart respectively. A total of three C# Scripts regulate the sub-function that takes place into these two **Unity Panels**.

- Dendrites_Item: Creates the content of a horizontal Scroll Panel that is a number of pre-created **Unity Buttons (UI)** equal to the number of neuron's dendrites.
- Synapses_Item: Creates the content of a horizontal Scroll Panel which (contains) is pre-created Texts with information about all the dendrite's full synapses (Synapse's ID, Kind, Groups and Source Neuron's ID).

- The final .cs file is called Plot. It is attached in both Neuron and Dendrite Panels. It is responsible for displaying the chart of the action potential for soma as well as neuron's dendrite. For the chart's line a **GameObject** is attached with a **Unity Line Renderer (Effect)**.

The main problem that had to be solved was that the two-dimensional **Canvas** has to be rotated along with the plot line which has to look stable. Initially, using the **Camera.main.WorldToScreenPoint** the screen's pixel from which the line has to start was found. Then, after every x and y of the line's screen position has been determined, with the use of the **Camera.main.ScreenToWorldPoint**, the **Line Renderer** is eventually designed into the three-dimensional World.

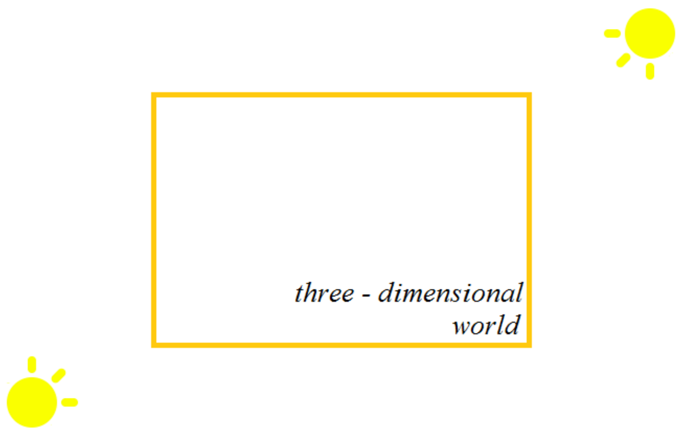
Camera

In this **Unity Scene** that is being described, the **Main Camera** is not static as it was in the previous two, but it is moving into space according to the user's mouse motion. This fact demands a Camera .cs file attached to it in order to manage its moving.

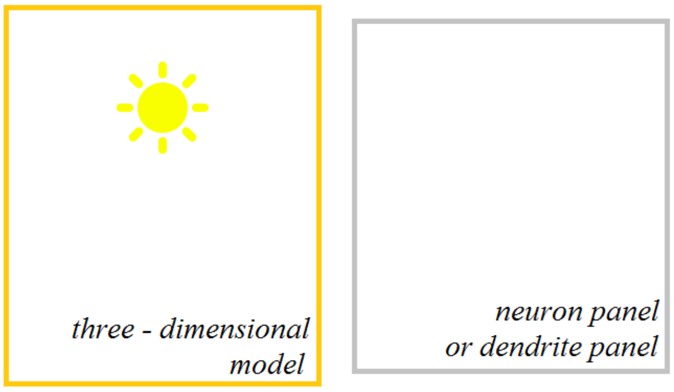
Specifically, into the .cs file there are two functions. The first one, regulates the camera moving in the three-dimensional space, whilst the second one regulates the rotation of the camera in case the Neuron or Dendrite Panel is active.

Light

For each one of the two modes (three-dimensional World or Neuron/Dendrite Panel is active), a different kind of lighting has been designed. In the first case, two **Unity Directional Lights** have been used to emit light into the three-dimensional point x,y,z = 0,0,0 from a proper distance. A **Unity Spotlight** emits light into the neuron-cell's model that is chosen in case that Neuron or Dendrite Panel is active.



Picture 5.15
Directional Lights

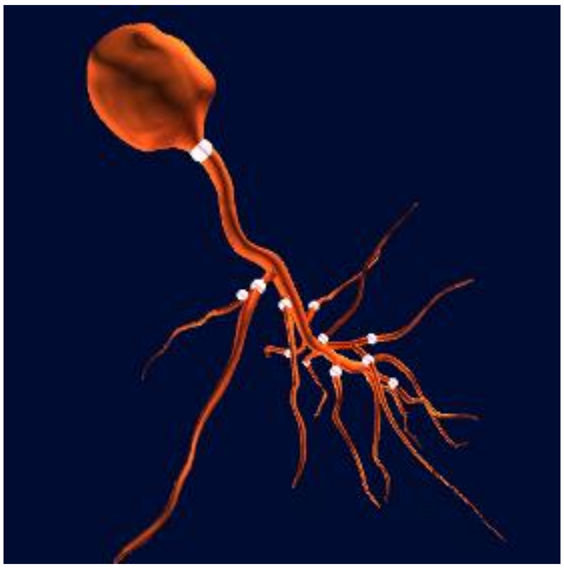


Picture 5.16
Spotlight

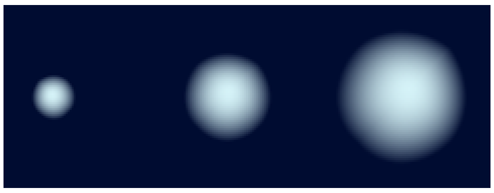
Lighting operation is regulated by the Windows_Handler .cs file. When the **Directional Lights** emit, the **Spotlight** is inactive, and vice versa.

The pre-created **Unity GameObject** that contains the three-dimensional models of the neuron-cell also contains the following:

- Three **Unity Halos (Effects)**: Each one of them has different size than the rest. The size indicates the action potential of the cell soma.
- Fourteen **Unity Spheres (3D)**: If the action potential exceeds the firing threshold, these Spheres are used in order to declare its propagation to the axon, and the axon terminals.
- The functions that were mentioned above are regulated by a C# Script named as Neuron_Firing.



Picture 5.17
Propagation of Action Potential
To Axon and Eventually to Axon Terminal
- Fourteen Spheres -



Picture 5.18
Neuron Soma's Action Potential
- Three-sized Unity Halos-

The dendrite three-dimensional model is contained in four different pre-created **Unity GameObjects**. In each one of them, the model has a different scale in order a neuron's dendrites to look dissimilar from each other.

Dendrite_Firing .cs file regulates the dendrite firing process by changing the material that is applied to the model.

This **Scene**, as the previous two also do, contains a Scenes_Manager .cs file that contributes to the transitions among the three **Unity Scenes**.

Chapter 6: Validation & Evaluation

6.1 Evaluation

Every application that has been implemented in the contents of a diploma thesis requires its operation validation and its evaluation by people that are directly related with that thesis object.

As it has already been mentioned, this application constitutes a user interface and a representation of the results of a neural network simulation. For its operation, an assist in validation by Computational Biology Lab (IMBB/FORTH) was required. A lab that is fully related with the implementation of the simulator and for over two months has in its position a beta version of the application.

Lab's member Athanasia Papoutsi (A.P.) has been asked to evaluate the application, by completing a form of evaluation consisting of eleven questions.

As it was mentioned in the first chapter, the purpose of the implementation of this application except for research, was the possibility of it beeing used as a study tool. For this reason, the biology professor of Secondary Education Christos Belbas (C.B.) also evaluated the application and after a quick but detailed demonstration, he also completed the form of evaluation.

Consequently, the answers of both of the evaluators are adduced.

Evaluation of the Application

- Interface -

Functionality

1. In the application the user is required to input a series of parameters. In your opinion, is the one provided the appropriate way to do it? Would an alternative way make that easier for the user? If so, what would that way be?

A.P.: The available way to provide parameters is in most cases the appropriate way. In the case of excitatory/ inhibitory neurons (page 1/2) it would be beneficial to provide the option of percentage of inhibitory neurons instead of absolute number.

C.B.: The provided way for parameter input is quite easy. It could also be based on the neuron's model representation though.

2. Is the camera in the three-dimensional form of visualization easy to handle?

A.P.: Yes, and it provides an excellent visualization of the data.

C.B.: Yes, sufficiently. Maybe a little more zoom could be useful.

3. Is the switching between the two tables in the two-dimensional form of visualization easy to handle?

A.P.: Yes, it is straightforward.

C.B.: Yes.

4. This application provides the user with three different forms of depiction of the data: three-dimensional models, chart and data tables. Do you believe that an extra form of data depiction is necessary? If so, which one would that be?

A.P.: No, I believe that these are adequate ways to visualize the data.

C.B.: No, it wouldn't be necessary. Perhaps it would be worth representing sub-cellular structures in the 3D model.

Appearance

5. From an aesthetic/artistic point of view, what is your opinion on the main menu's and two-dimensional's presentation? Would you alter/improve something?

A.P.: Yes. For the two-dimensional data I would plot a yes or no event (pass the threshold or not) with a bar, instead of the amplitude value. In this way a more concrete visualization of the data could be achieved. For amplitude details, one could switch to the 3D visualization.

C.B.: From an aesthetic point of view it is quite good. The default settings may need some improvement.

6. What impression does the choice of colors leaves you with? Is there the proper contrast in order the displaying of the components to be distinct?

A.P.: The colors used in the 3D visualization are the appropriate ones and one can distinct the relevant parameters. For the 2D visualization, a higher contrast may be easier to see.

C.B.: The choice of colors is proper given that it reflects the different states of displaying the action potential. Maybe some more vivid colors could be used.

7. What is your opinion on the three-dimensional models of the neurons and dendrites? Are they aesthetic and realistic? Could they be improved?

A.P.: The visualization of the neurons and dendrites are aesthetic and realistic.

C.B.: They are very well designed. In order for them to be more realistic, the cell models should be a little bit closer to each other.

8. Do you believe that the representation of the firing in soma and dendrite layer is satisfactory?

A.P.: Yes, one can easily see when a neuron/dendrite are firing.

C.B.: Yes. Maybe a sub-cellular representation of the firing events would be necessary.

- As an Educational Tool -

9. In your opinion, could this application – with potential alterations- be used as a tool to assist the understanding of the neuron's function from an educational organization? For instance, could it be used from a school in order to enable its students to observe the nervous system's operation?

A.P.: Yes, in my opinion, this is a really nice tool that could engage students and help their understanding in neuronal function.

C.B.: Yes, sure. This application will of course be used as a base of an educational program. Moreover, it should correspond with the analytical school program. In the first class of high school it could definitely be used for the teaching of the nervous system.

10. What alterations (additions – removals) would you suggest in this case?

A.P.: None.

C.B.: The parameters should be less, their input should be based on the 3D models, it should contain information on sub-cellular structures and procedures, such as the nerve pulse.

- Comments -

11. What is your overall impression of this application? Any observations/ conclusions?

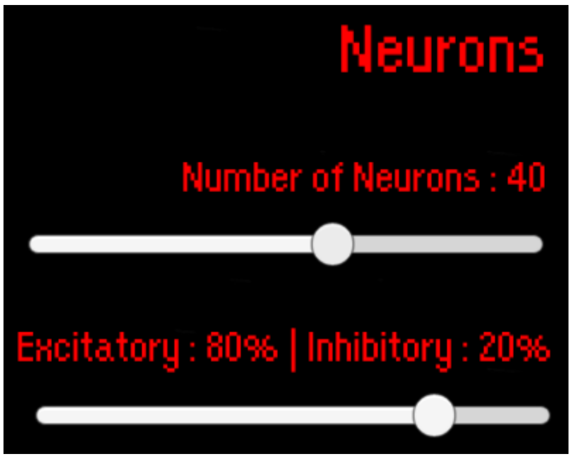
A.P.: This is an excellent graphical interface for data visualization.
Improvements that I would suggest is
a) provide the option to run many times the same simulation (given random processes)
b) Provide an option to define the number of inhibitory/excitatory synapses
Overall, this is a high-standard work, and provides a useful tool for neuroscientists.

C.B.: Overall it's a quite good application as a base. Thereon, the continuing of this particular project is related to the purpose that emerges. If it should support an educational process, a different development is required, likewise if it should be utilized as a part of a broader research project. However, even in its current form, with the help of a meticulous manual it could be used online for educational purposes.

6.2 Improvements

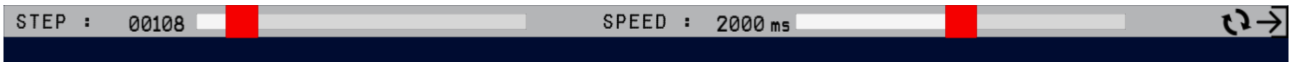
The responses of the evaluators constitute an important factor for useful conclusions about the application. Some of their comments were incorporated as corrections / improvements in the implemented application as it was described in Chapters 5 and 6. These improvements are listed below.

According to A.P.'s answer in Q1, in the first out of two Input of Parameters panel, in the Neurons section the user is enabled to regulate the option of percentage of inhibitory neurons instead of absolute number.



Picture 6.1 (improved Picture 4.4.1)
Neurons Parameters

According to C.B.'s answer in Q5, the speed of the visualization can be chosen from half of a second to three and a half seconds, with the default value equals with two seconds.



Picture 6.2 (improved Picture 4.15)
Hidden Task Bar

Chapter 7: Conclusion

7.1 Comments

In this diploma Thesis an application for the visualization of a neural network was implemented. The user is enabled to insert the desirable parameters in order to execute a simulation and after it has finished, the user has the ability to watch its results in two different forms. The first one was based on a table that contains the action potential for the neurons and for dendrites. The second form was a representation of a three-dimensional world with three-dimensional cell models, and also a graph for neuron's action potential.

Specifications:

CPU	Intel(R) Core(TM) i7-3632QM CPU @2.20 GHz (4 CPUs)
RAM	8 GB
OS	MS Windows 8.1 Pro x64, MS Windows 10 Education x64
Video Card	AMD Radeon HD 7500M/7600M Series

Software versions:

Unity	Unity 5.1.x Personal, Unity 5.3.x Professional
MS Visual Studio	MS Visual Studio Ultimate 2013, MS Visual Studio Enterprice 2015

Blender	Blender 2.7x
Autodesk 3ds Max	Autodesk 3ds Max 2015
Autodesk Mudbox	Autodesk Mudbox 2015
MS Visio	Visio Professional 2013 SP1, Visio Professional 2016

Picture 7.1
Specification & Software Versions

To measure the efficiency of this application in terms of graphic display, the frames per second rate was counted. In the current system with the specifications that were described above, the following results were observed.

In Scene 0, the Input Data Menu, the measured rate was steadily at 60 fps.

In Scene 1, the Tables of Neurons and Dendrites, in order to test the application's efficiency, the parameters were set in their maximum values (60 neurons and 64 dendrites). In that case the rate was continuously above 30 fps.

For the second Scene, the 3D Space Visualization, it was expected that the fps rate would the lowest when the rendering of the 3D world space in its entirety (total number of simulation's neurons and dendrites) would be requested. For that reason, the efficiency was tested in a number of different combinations of simulation parameters, such as:

- 60 neurons (max) - 100% excitatory - 64 dendrites per neuron (max)
  $60 + 60 * 64 = 3900$ 3D models => 14 fps.
- 60 neurons (max) - 80% excitatory and 20% inhibitory - 64 dendrites per neuron (max)
  $60 + 48 * 64 + 12 * 1 = 3144$ 3D models => 18 fps.
- 60 neurons (max) - 100% excitatory - 34 dendrites per neuron (mean)
  $60 + 60 * 34 = 2100$ 3D models => 20 fps.
- 60 neurons (max) - 80% excitatory and 20% inhibitory - 34 dendrites per neuron (mean)
  $60 + 48 * 34 + 12 * 1 = 1704$ 3D models => 24 fps.

From these results it is concluded that even in cases of 14 and 18 fps, which are considered as a low rate, the user has no difficulty to interact with the application, let alone it is impossible to lose pieces of information. Alongside, the fact that it is almost improbable to use these specific parameters in a real simulation, reinforces the fact that the low fps rate does not constitute a problem.

However, whenever the user performs the main functions in Scene 2, like moving through the 3D space among the neurons or displaying a neuron or dendrite panel, the fps rate is high (in the order of 50 to 60 fps) something that makes the previous results rather unimportant.

It should also be noted that parts of the design such as plots (Scene 2) and tables (Scene 1) were consciously chosen to be represented per hundred simulation steps, in order to leave unaffected the number of fps, thus greater number of fps rate would cause a malfunction of the application and hardship in its management by the user.

7.2 Future Work

The main problem that the user has to deal with, constitutes the fact that, as a standalone application, an interaction with the server that hosts the emulator is required. The uploading and downloading of the necessary files is a time-consuming process, given that the time of downloading the files after the simulation is proportional to the chosen simulation time.

The necessary steps for improvement on this issue took place, but the size of the problem remained noticeable. The best way to avoid this challenge and complete its elimination, is the conversion of the application into a web one, something that required little effort, nevertheless wasn’t implemented due to lack of available time in the contents of this diploma thesis. This Web application, as it will be hosted in the same server, will not require the downloading or the uploading of any file.

Concerning the future work and the continuing of this particular project, firstly the purpose that emerges must be clear, as it was referred by professor Christos Belbas. Though the initial purpose of this application's creation was to frame the simulation's algorithm in order to act as a complete tool in order to be used by neuroscientists, with minimal changes could frame the teaching of the nervous system in a school organization.

In this case, always in accordance with the recommendations of the expert, a reduction of the parameters the user must insert as well as an introduction of an attempt to represent the processes occurring within the cell such as containing information on sub-cellular structures and procedures like the nerve pulse, would be necessary.

In opposition to the previous, the development and improvement of a comprehensive tool for the simulation and the representation of a neural network using by scientists, would require the changes that were introduced by the member of the Computational Biology Lab of IMBB / FORTH Athanasia Papoutsi. Specifically, the application has to provide the user the option to run the same simulation many times (with random processes) as well as an option to define the number of inhibitory/excitatory synapses.

In the case of repeated simulations, a good addition would be the ability of the user to delete - during the visualization by pressing a button - a particular neuron and/or a dendrite that he does not desire it to be contained in the next simulation.

Along with the suggestions of experts, there are suggestions that would improve the implementation of both the operating level and display level.

A necessary addition which should be implemented is the representation of the interconnection between the cells as well as the representation of the pulse's transmission from the presynaptic to the postsynaptic neuron.

Also, a necessary addition would be providing the ability to the user to choose a third form of visualization, that would be - in combination with the MathWorks MATLAB tool - a raster plot of the activity of the neurons with the option to respectively display a raster plot of their dendrites′ activity.

References

[1]. Y CAJAL, Santiago Ramón. Histologie du système nerveux de l'homme & des vertébrés, vol. 1, Paris : Maloine, 1909, p. 1014.
[2]. https://en.wikipedia.org/wiki/Purkinje_cell
[3]. HODGKIN, Alan L.; HUXLEY, Andrew F. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of physiology, 1952, 117.4: 500.
[4]. https://unity3d.com/
[5]. KOUSANAKIS, Manolis. Neural Network Simulation Speedup based on Reconfigurable Logic.2015.
[6]. http://www.dendrites.gr/
[7]. http://www.erasmusbrainproject.com/
[8]. https://phet.colorado.edu/
[9]. http://www.artificialbrains.com/digicortex
[10]. http://www.neurogems.org/neosim2/index.html
[11]. BEULER, Marcel, et al. Real-Time simulations of synchronization in a conductance-based neuronal network with a digital FPGA hardware-core. In: Artificial Neural Networks and Machine Learning–ICANN 2012. Springer Berlin Heidelberg, 2012. p. 97-104.
[12]. KASIŃSKI, Andrzej; PAWŁOWSKI, Juliusz; PONULAK, Filip. 'SNN3DViewer'-3D Visualization Tool for Spiking Neural Network Analysis. In: Computer Vision and Graphics. Springer Berlin Heidelberg, 2008. p. 469-476.
[13]. Crazy Eddie's GUI SystemMk-2, http://www.cegui.org.uk, Copyright c 2004 – 2005 CEGUI Team & Contributing Authors
[14]. https://msdn.microsoft.com/en-us/library/windows/desktop/bb153256%28v=vs.85%29.aspx
[15]. VON KAPRI, Anette, et al. Towards the visualization of spiking neurons in virtual reality. In: MMVR. 2011. p. 685-687.
[16]. GLEESON, Padraig; STEUBER, Volker; SILVER, R. Angus. neuroConstruct: a tool for modeling networks of neurons in 3D space. Neuron, 2007, 54.2: 219-235
[17]. CROOK, Sharon, et al. MorphML: Level 1 of the NeuroML standards for neuronal morphology data and model specification. Neuroinformatics, 2007, 5.2: 96-104.
[18]. HINES, Michael L.; CARNEVALE, Nicholas T. The NEURON simulation environment. Neural computation, 1997, 9.6: 1179-1209.
[19]. BOWER, James M.; BEEMAN, David. The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System. Springer Science & Business Media, 2012.
[20]. http://www.reson8.com.au/neuron-matrix/#!prettyPhoto
[21]. http://nxxcxx.github.io/Neural-Network/
[22]. http://wikis.ece.iastate.edu/cpre584/images/6/64/ConveyPDKReferenceManual.pdf
[23]. http://hdwyn.com/wallpaper/black_background_red_color_paint_explosion_hd-wallpaper-9542.jpg
[24]. https://sshnet.codeplex.com/
[25]. http://entitycrisis.blogspot.gr/2011/02/uniform-points-on-sphere.html