# Hanitu User Guide

## For Hanitu V.1.3-r8

## 2015.8

**Chung-Chuan Lo Laboratory**
Institute of Systems Neuroscience
National Tsing Hua University, Taiwan

# Table of contents

# 1. Introduction

## 1.1. What is Hanitu

Hanitu is a simulation environment that simulates behavior and neural activity of user-designed virtual worms in a two-dimensional virtual world. In Hanitu, the users need to think deeply about how different components of a nervous system work together to achieve the ultimate goal of an animal – to survive in a changing environment

Hanitu is designed for graduate and undergraduate students who already acquired basic knowledge about computational neuroscience and would like to apply what they learned to addressing some of the system-level questions in neuroscience.

## 1.2. System requirements

For general users: The current version of Hanitu is compatible with and has been tested on Ubuntu Linux. The Hanitu software package comes with pre-compiled executables for Linux and contains Java SE Runtime Environment (java 1.8) already.

For developers: Hanitu (the virtual world) and Flysim (the simulator) are written in C++ with C++11 library, compiled with gcc 4.8. The GUI is written in Java.

## 1.3. Installation

The Hanitu system does not need installation. The users only need to download the entire contains of the software package (32-bit or 64-bit) into a local hard drive. The users many need to set the program files (*.out) as executable if they are not (using the “chmod” command).

## 1.4. The content in the Hanitu software package

The Hanitu software package contains the following top-level directories:

- **bin**：Containing scripts for different functions. The users can run it through terminal.
- **jre1.8**：Java Runtime Environment (JRE).
- **lib**：Containing precompiled executable program files of Hanitu (Hanitu.out) and Flysim (flysim.out) and compressed package of GUI interface software (HanituToolSet.jar).
- **share**：Containing three folders：
  - script：Sample script files for running Hanitu or for data analysis.
  - sample：Sample circuit (*.ccg) and virtual world (*.wcg) configuration files.
  - documents：User guide, manual or other documents.

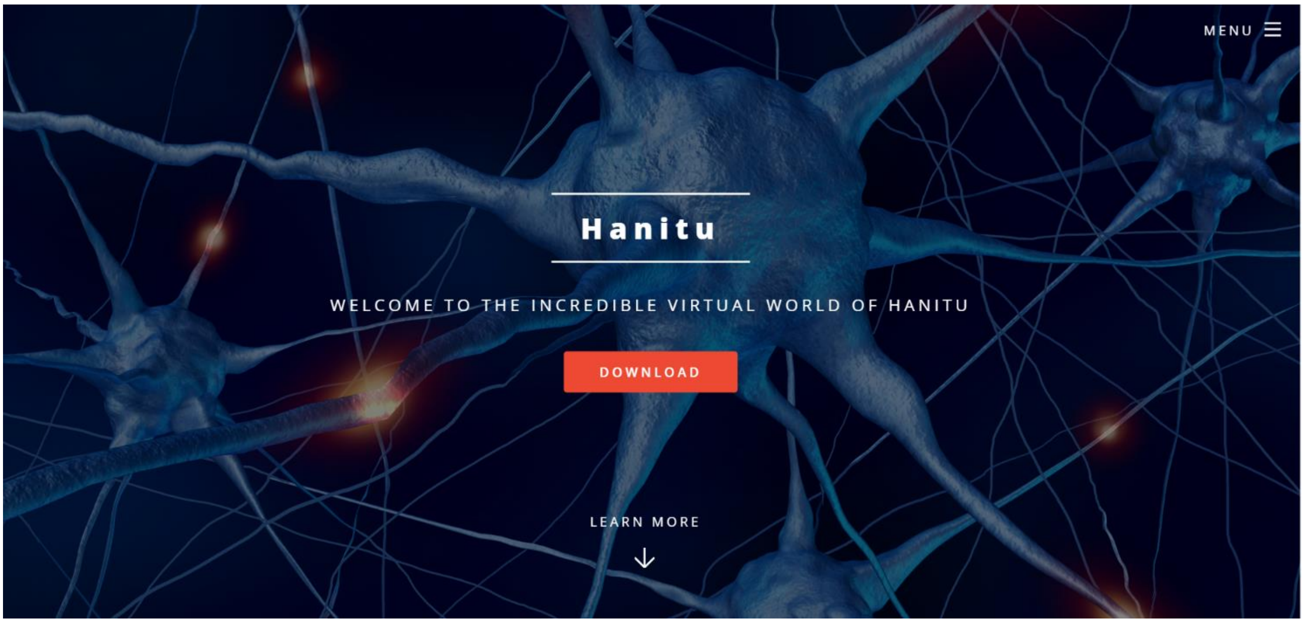
The Hanitu software package contains several executables, scripts and configuration files:

- **HanituGUI**：**The main program users need to run.** The users can run it from the terminal or by clicking on it in a file manager. All the major functions of Hanitu (virtual world design, circuit design, simulation and virtual world visualization) are included in this GUI.
- **bin/hanitu_world**：The script brings up a GUI for setting virtual world configuration. See 3.1.GUI for virtual world configuration setting for GUI description. This script is integrated into HanituGUI, but users can run the script independently if they need to.
- **bin/hanitu_circuit**：The script brings up a GUI for designing virtual worm circuits. See 3.2.GUI for designing virtual worm for GUI description. This script is integrated into HanituGUI, but users can run the script independently if they need to.
- **bin/hanitu_plot**：This script brings up a GUI and display movement of the worms by reading Locations.txt (See 4.1.Locations.txt) that was saved previously. See 3.3.GUI for display for GUI description. This script is integrated into HanituGUI, but users can run the script independently if they need to.
- **bin/hanitu_run**：Run Hanitu.out, Flysim.out and plot. This program is integrated in HanituGUI. But if the users want to skip world design and circuit design by performing the simulation directly, they can execute this program.

- **bin/hanitu_concentration**：The script calculates single molecule diffusion concentration of food or toxicants in different distance. This script is not included in HanituGUI.
- **bin/hanitu_filter**：The script is used for post-simulation analysis. It extracts information from Spike.txt (See 4.2.Spike.txt) such as spiking time and neuron ID according to specific user ID, virtual worm ID and type of neurons(sensory、motor、body). This script is not included in HanituGUI.
- **bin/hanitu_firingrate**：The script is used for post-simulation analysis in conjunction with hanitu_filter. It calculates firing rate based on the output data generated by hanitu_filter. This script is not included in HanituGUI.
- **bin/hanitu_tool**：The main script called by HanituGUI. Users do not need to interact with this script.
- **lib/Hanitu.out**：The main program of Hanitu. The users do not need to interact with the program.
- **lib/flysim.out**：The neural network simulator used in Hanitu. The users do not need to interact with the program.
- **share/script/template.sh**：A sample script for post-simulation analysis.
- **share/sample/4neurons.ccg**、**direct_connection.ccg**、**simple_decision.ccg**：Three sample worm configuration files. Each file contains parameters used to define the neural circuit of a worm species. The users can edit files directly. See Appendix A.2.Circuit configuration files for details.
- **share/sample/world_config.wcg**：A sample configuration file for the virtual world. See Appendix A.1. Virtual world configuration file for details.
- **Hanitu Simple Guide V.1.3.pdf**：This Hanitu user guide.

## 1.5. Official website

The official website is at http://hanitu.net.



# 2. How to run Hanitu

## 2.1. Using GUI

1. The users can use the GUI (HanituGUI) to edit virtual world configuration files (*.wcg, see Appendix A.1.Virtual world configuration file for details) and virtual worm circuit configuration files (.cgg, see Appendix A.2.Circuit configuration files for details). The users can also use the GUI to lunch simulations and view the virtual world with the worm survival games in action. There are three methods for executing:

   a. Double-click on the HanituGUI icon in a file manager.

   b. Enter following command in the terminal[1]

   ```
   ./HanituGUI
   ```

   c. Move to "bin" folder and enter command below under the ./bin subdirectory in the terminal

   ```
   ./hanitu_world [FILE]
   ```

   [FILE] should point to a world configuration file.

2. To perform simulations and to display virtual worm movements only, simply enter the following command under the ./bin subdirectory

   ```
   ./hanitu_run [FILE]
   ```

   [FILE] should point to a world configuration file.
   The users need to make sure that the virtual world configuration file (.wcg，default is world_config.wcg) and the corresponding virtual worm circuit configuration files (.ccg) are presented in the same directory.

3. The users can enter the following command to execute GUI for only displaying virtual worm movements based on the data previously stored in a location files (See 4.1.Locations.txt).

   ```
   ./hanitu_plot [Location file]
   ```

---

[1] The users can get help information through adding "`-h`" behind the commands.

## 2.2. Performing Hanitu simulations directly (without GUI)

First, open two terminals. Run Flysim (under /lib) in one terminal[2]:

```
./flysim.out -daemon 8889
```

and run Hanitu (under /lib) in the other terminal:

```
./Hanitu.out -w [world config FILE]
```

The users need to make sure that virtual world configuration file (.wcg，default is world_config.wcg) and corresponding virtual worm circuit configuration files (.ccg) are presented in the same directory.

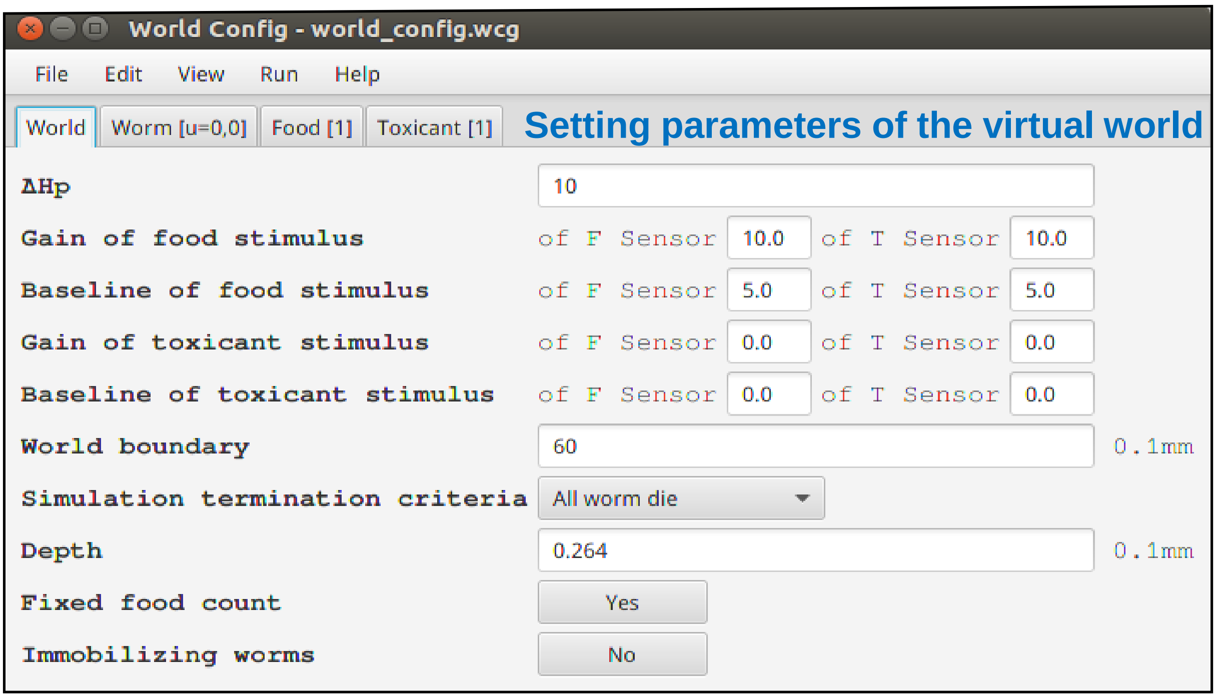
---

[2] The users can get more function information by adding “`-h`” behind commands.

# 3. How to use the GUI

## 3.1. Setting the virtual world in the main GUI

After executing HanituGUI, the users will see the following main GUI which is used for setting parameters of the virtual world and related parameters.



Setting parameters of the virtual world

ΔHp: 10

Gain of food stimulus: of F Sensor 10.0, of T Sensor 10.0

Baseline of food stimulus: of F Sensor 5.0, of T Sensor 5.0

Gain of toxicant stimulus: of F Sensor 0.0, of T Sensor 0.0

Baseline of toxicant stimulus: of F Sensor 0.0, of T Sensor 0.0

World boundary: 60 0.1mm

Simulation termination criteria: All worm die

Depth: 0.264 0.1mm

Fixed food count: Yes

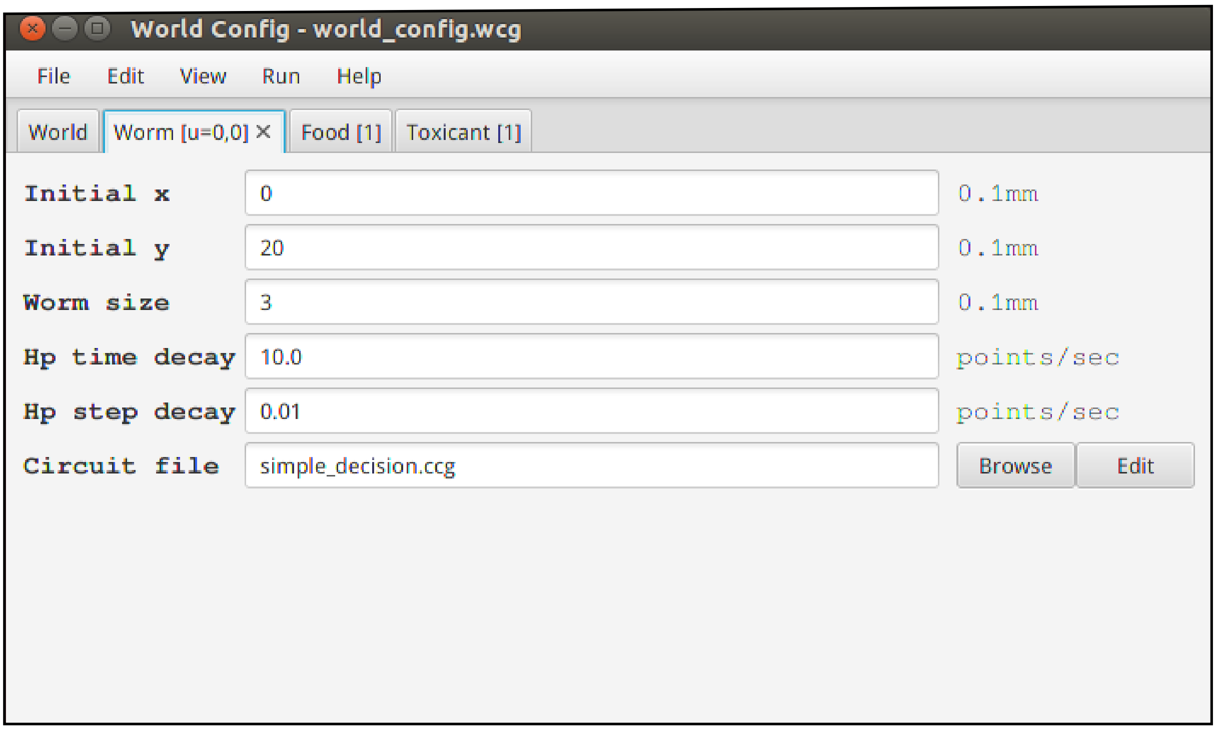
Immobilizing worms: No

| **Parameters in GUI** | **Corresponding parameters in configuration files** | **Description** |
|---|---|---|
| **ΔHp** | Nutrient | The energy level increased or decreased when a worm touches a food or a toxicant, respectively |
| **Gain of food stimulus of F Sensor** | TransformA | Define the relation between the sensory neuron input rate and the odorant concentration for the food stimuli with respect to the food sensory neurons. TransformA defines the gain and TransformB defines the baseline[3] . |
| **Baseline of food stimulus of F Sensor** | TransformB | |

---

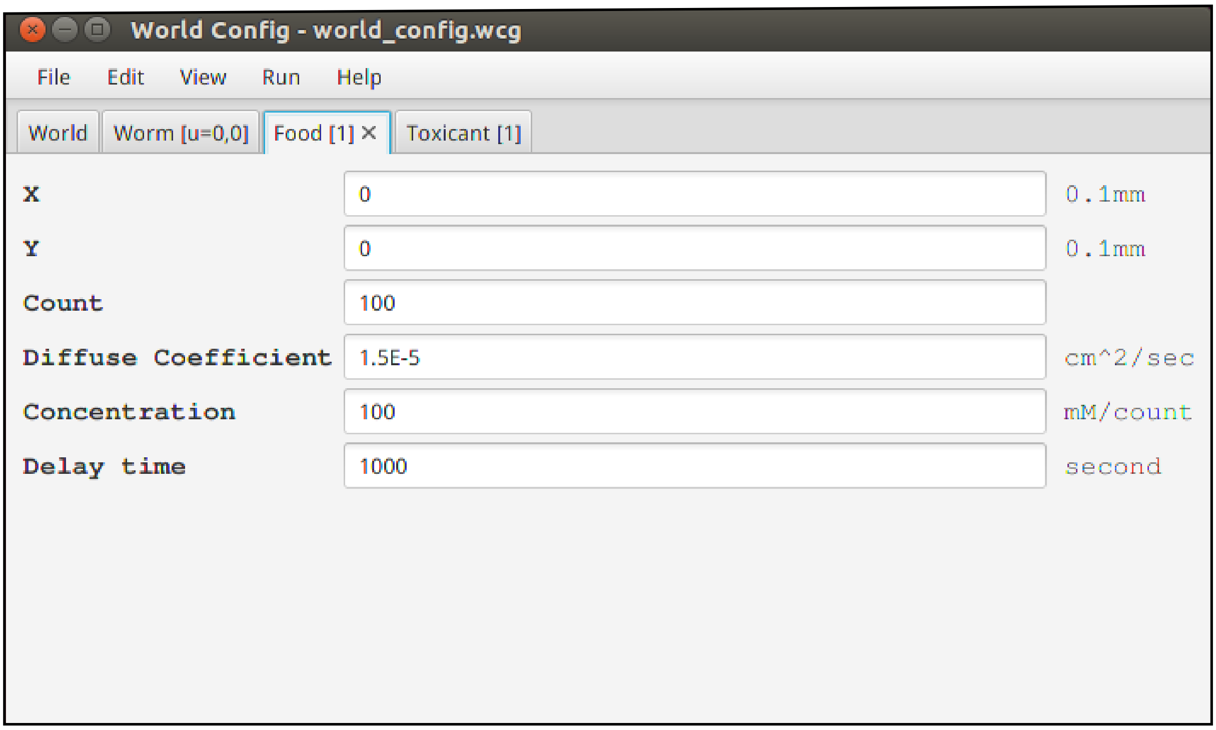
[3] **Equation**：$\text{firing rate(Hz)} = Gain \times \ [\text{Odor}](\text{mM}) + \text{Baseline},$ $[\text{Odor}]$ means concentration of molecule $(\text{mM})$.

| Parameters in GUI | Corresponding parameters in configuration files | Description |
|---|---|---|
| **Gain of food stimulus of T Sensor** | TransformA_FT | Similar to TransformA and TransformB but for the food stimuli with respect to the toxicant sensory neurons |
| **Baseline of food stimulus of T Sensor** | TransformB_FT | |
| **Gain of toxicant stimulus of F Sensor** | TransformA_TF | Similar to TransformA and TransformB but for the toxicant stimuli with respect to the food sensory neurons |
| **Baseline of toxicant stimulus of F Sensor** | TransformB_TF | |
| **Gain of toxicant stimulus of T Sensor** | TransformA_TT | Similar to TransformA and TransformB but for the toxicant stimuli with respect to the toxicant sensory neurons |
| **Baseline of toxicant stimulus of T Sensor** | TransformB_TT | |
| **World boundary** | Boundary | The boundary (between -50 and 50 for both x and y) of the virtual world. The unit is 0.1mm. |
| **Stimulation termination criteria** | Type | Criteria for the simulation termination. Type=0, terminated when all worms die. Type=1, terminated when a worm first touches a food or a poison. |
| **Depth** | Depth | Height of the world. Used for calculating the odorant concentration. Do not alter. |
| **Fixed food count** | CountMode | Do not change this parameter. It tells the program that the amount of food in each food source is unlimited. |
| **Immobilizing worms** | Fixed | Fixing location of virtual worm or not. 1 for fixing and 0 for not. A worm with fixed location is useful for the testing purpose. |



Initial x: 0 0.1mm
Initial y: 20 0.1mm
Worm size: 3 0.1mm
Hp time decay: 10.0 points/sec
Hp step decay: 0.01 points/sec
Circuit file: simple_decision.ccg

| **Parameters in GUI** | **Corresponding parameters in configuration files** | **Description** |
|---|---|---|
| Initial x | InitialX | Worm's initial location in x and y. The unit is one step (0.1mm) of the worm. |
| Initial y | InitialY | |
| Worm size | Wormsize | Radius of the worm. (0.1mm) |
| Hp time decay | Time_decay | Energy level decay per second. (Initial value = 100) |
| Hp step decay | Step_decay | Energy level decay per step of the worm |
| Circuit file | Filename | The circuit configuration file of the worm |



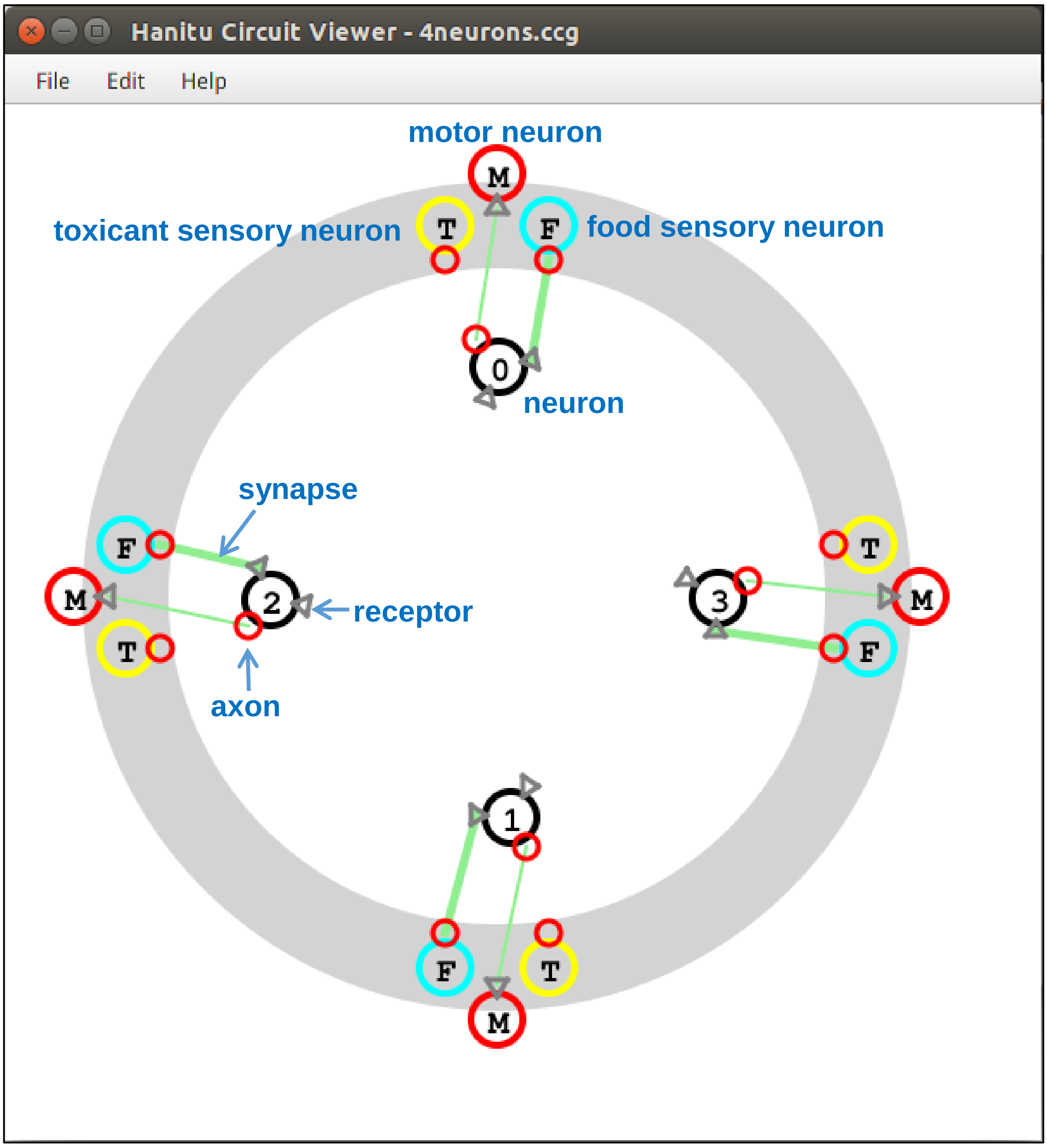
X 0 0.1mm
Y 0 0.1mm
Count 100
Diffuse Coefficient 1.5E-5 cm^2/sec
Concentration 100 mM/count
Delay time 1000 second

| Parameters in GUI | Corresponding parameters in configuration files | Description |
|---|---|---|
| X | X | The location (in 0.1 mm) of the food source |
| Y | Y | |
| Count | Count | The amount of foods in each food source |
| Diffuse Coefficient | Diffuse | The diffusion constant of the food molecule $cm^2/sec$ |
| Concentration | Concentration | The concentration of the food molecule (mM/count) at the food source for each food count. |
| Delay time | Delay_time | The virtual worm simulation starts with a delay after the foods are placed in the assigned locations. This parameter specified the delay time (in 0.1 msec) |

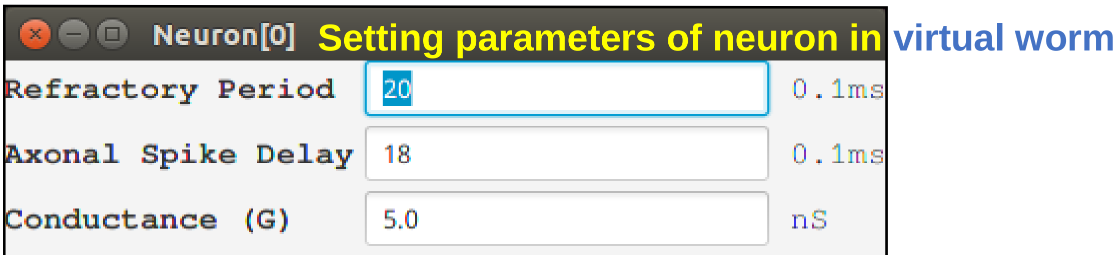
The GUI and parameters for toxicants are the same with those for foods.

## 3.2. Designing the virtual worm neural circuits

The user can edit the neural circuit of a giving worm in a GUI shown below. The GUI can be brought up by clicking the “Edit” button in a “Worm” tab in the virtual world GUI described in 3.1. In this GUI, the users can add new neurons by right-clicking. The small red circle on each neuron indicates an outgoing axon and the triangle indicates a receptor targeted by the axon from an upstream neuron. The users can connect two neurons by left-clicking on a red circuit of a source neuron and right-clicking on a triangle target on a target neuron.



The users can set parameters of a neuron by selecting (left-click) it and press “I”, which brings up a GUI as shown below.

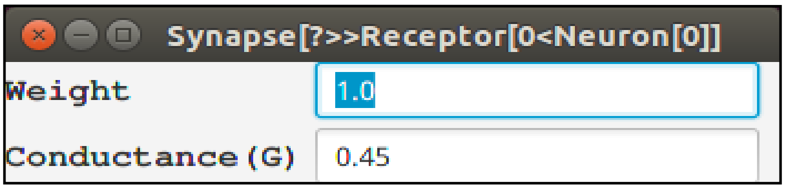


Refractory Period 20 0.1ms

Axonal Spike Delay 18 0.1ms

Conductance (G) 5.0 nS

| **Parameters in GUI** | **Corresponding parameters in configuration files** | **Description** |
|---|---|---|
| Refractory Period | Refperiod | Refractory period (timestep = 0.1 ms) |
| Axonal Spike Delay | Spikedelay | Spike delay (timestep = 0.1 ms) |
| Conductance (G) | G | Membrane leak conductance (ns) |
| Capacitance (C) | C | Membrane capacitance (nF) |
| Reversal Potential | NRevPot | Membrane reversal potential (mV) |
| Reset Potential | ResetPot | Reset Potential (mV) |
| Spike Threshold | Threshold | Spike threshold (mV) |
| Noise Std | std | Standard deviation of the membrane noise (current). |
| Noise Mean | mean | Mean of the membrane noise (current). |



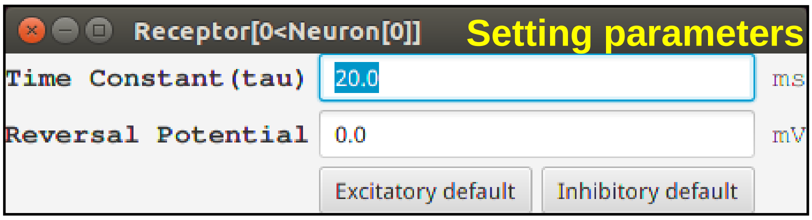
Weight 1.0

Conductance (G) 0.45

## Setting parameters of synapse

| **Parameters in GUI** | **Corresponding parameters in configuration files** | **Description** |
|---|---|---|
| Weight | Weight | Synaptic strength = Weight x G(ns) |
| Conductance (G) | G | |

## Setting parameters of receptor

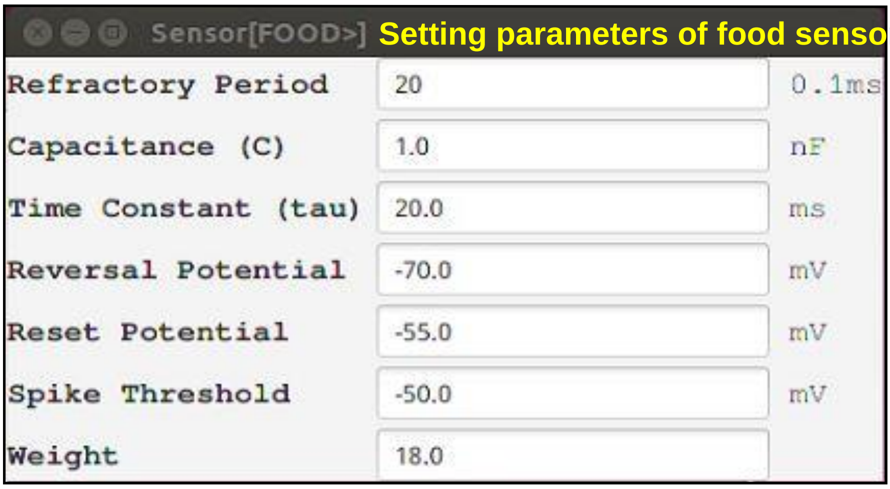


Time Constant(tau) 20.0 ms

Reversal Potential 0.0 mV

| **Parameters in GUI** | **Corresponding parameters in configuration files** | **Description** |
|---|---|---|
| Time Constant (tau) | Tau | Time constant (ms) of the synaptic receptor |
| Reversal Potential | RRevPot | Reversal potential (mV) of the synaptic receptor |

| **Default parameters for the excitatory and inhibitory neurons** | | |
|---|---|---|
| **Parameters in GUI** | **Excitatory** | **Inhibitory** |
| Time Constant (tau) | 20.0 ms | 20.0 ms |
| Reversal Potential | 0.0 mV | -70.0 mV |

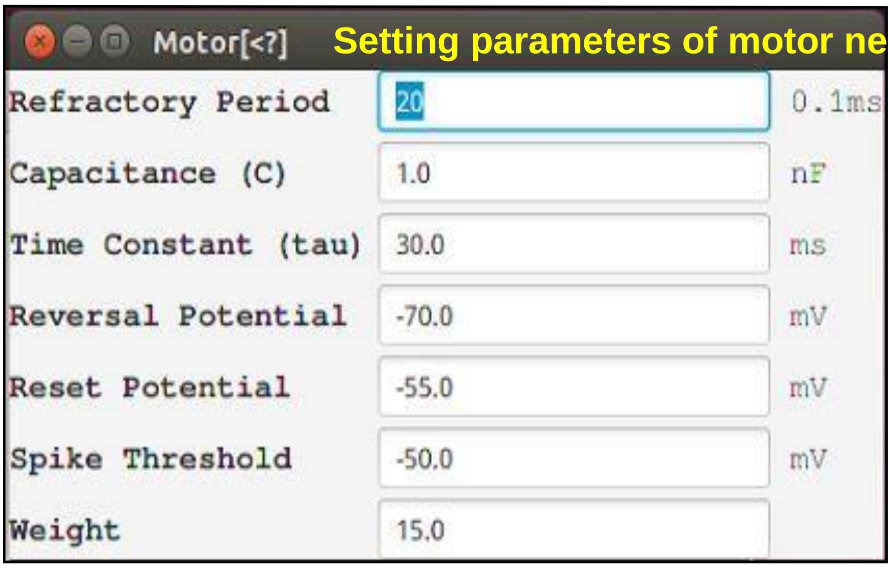


Setting parameters of food sensory neuron

Refractory Period: 20 0.1ms
Capacitance (C): 1.0 nF
Time Constant (tau): 20.0 ms
Reversal Potential: -70.0 mV
Reset Potential: -55.0 mV
Spike Threshold: -50.0 mV
Weight: 18.0

| **Parameters in GUI** | **Corresponding parameters in configuration files** | | **Description** |
|---|---|---|---|
| Refractory Period | F | SFsilence | Neuron refractory period (0.1 ms) |
| | T | SCsilence | |
| Capacitance (C) | F | SFcm | Membrane capacitance (nF) |
| | T | SCcm | |
| Time Constant (tau) | F | SFtau | Synaptic time constant (ms) |
| | T | SCtau | |
| Reversal Potential | F | SFvl | Membrane resting potential (mV) |
| | T | SCvl | |
| Reset Potential | F | SFreset | Reset Potential (mV) |
| | T | SCreset | |
| Spike Threshold | F | SFvth | Spike threshold (mV) |
| | T | SCvth | |
| Weight | F | Weight<br>(In Inputneuron part) | Synaptic weight |
| | T | | |

The GUI used to set parameters of toxicant sensory neuron is similar to that for food sensory neurons. Abbreviations: “F” for food sensory neurons and “T” for toxicant sensory neurons.

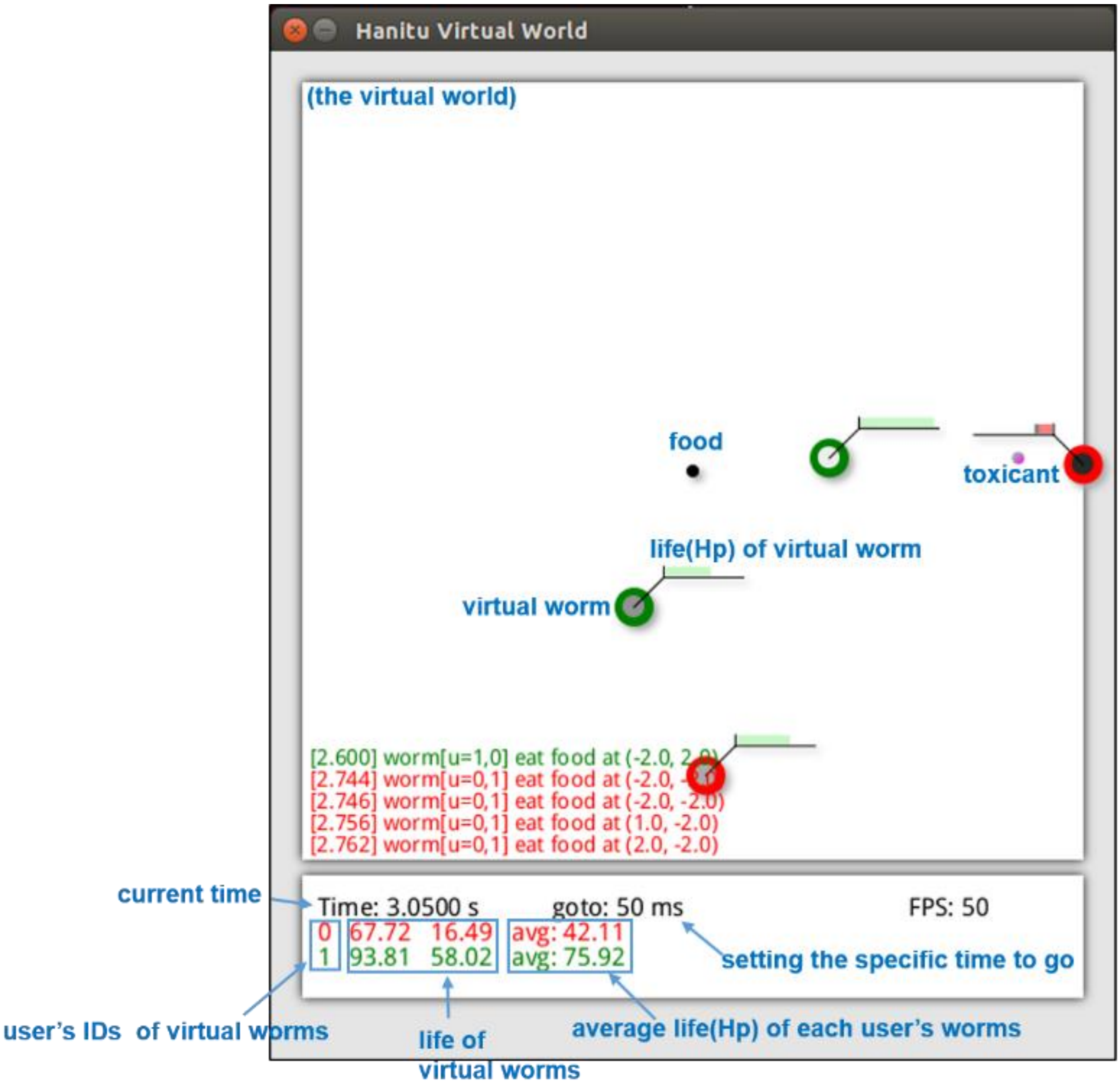


Setting parameters of motor neuron

Refractory Period 20 0.1ms
Capacitance (C) 1.0 nF
Time Constant (tau) 30.0 ms
Reversal Potential -70.0 mV
Reset Potential -55.0 mV
Spike Threshold -50.0 mV
Weight 15.0

| Parameters in GUI | Corresponding parameters in configuration files | Description |
|---|---|---|
| Refractory Period | Msilence | Neuron refractory period (0.1 ms) |
| Capacitance (C) | Mcm | Membrane capacitance (nF) |
| Time Constant (tau) | Mtau | Synaptic time constant (ms) |
| Reversal Potential | Mvl | Membrane resting potential (mV) |
| Reset Potential | Mreset | Reset Potential (mV) |
| Spike Threshold | Mvth | Spike threshold (mV) |
| Weight | Mweight | Synaptic weight |

## 3.3. Viewing the simulation



```
[2.600] worm[u=1,0] eat food at (-2.0, 2.0)
[2.744] worm[u=0,1] eat food at (-2.0, -
[2.746] worm[u=0,1] eat food at (-2.0, -2.0)
[2.756] worm[u=0,1] eat food at (1.0, -2.0)
[2.762] worm[u=0,1] eat food at (2.0, -2.0)
```

```
Time: 3.0500 s        goto: 50 ms            FPS: 50
0  67.72  16.49  avg: 42.11
1  93.81  58.02  avg: 75.92
```

After clicking “Run Hanitu and plot” in the “Run” menu of the main GUI, the users bring up a panel (see above) which shows an animation for the virtual worm simulation. Note that the worm movements shown in the GUI is not in real time because the simulation is often much faster than what the GUI shows. The circles represent the worms with horizontal bars indicating the Hp value. The black dots indicate the food sources and the magenta dots are toxicant sources. When a worm eats foods or hits toxicants, the information is displayed at the lower-left corner of the virtual world. The users can press the space button to pause the animation and PgUp/PgDn to change FPS (frame rate per second). The users can enter a number (in msec) then press the enter button to switch to the specified time point.

# 4. Output files

The Hanitu system has two output files:

## 4.1. Locations.txt

The file records the locations(X, Y) and energy levels(Hp) of all worm. The format is as follows:

```
Time(ms)   UserID     WormID     X      Y      Hp
0          0          0          10     -2     99.3599
```

## 4.2. Spike.txt

The file records spike times of all neurons from all worms. The format is as follows:

```
Time(0.1ms) UserID      WormID      NeuronID        NeuronType
22.7        0           0           3               s
```

There are three types of neuron in the Hanitu system: s for sensory, m for motor and b for body.

# 5. Model and Simulations

## 5.1. Neuron model

Neuron model Hanitu supports leaky integrate-and-fire model:

$$C_m \frac{dV}{dt} = g_L(V - E_L) + I_{syn},$$

where $C_m$(nF) is the membrane capacitance, $V$(mV) is the membrane potential, $g_L$(ns) is the membrane leak conductance, $E_L$(mV) is membrane reversal potential and $I_{syn}$(nA) is the synaptic current. Spike threshold, reset potential, refractory period and spike delay are also implemented in the neuron model. All the parameters are user tunable.

| Parameters of model | Parameters in GUI | Parameters in the circuit file |
|---|---|---|
| $\boldsymbol{C_m}$ | Capacitance (C) | C, SFcm, SCcm, Mcm |
| $\boldsymbol{g_L}$ | Conductance (G)<br>(In body neuron part) | G |
| $\boldsymbol{E_L}$ | Reversal Potential | NRevPot, SFvl, SCvl, Mvl |

Note: In the sensory neurons and motor neurons, the membrane leak conductance is set to a fixed value of 2.5 (ns).

## 5.2. Synapse model

Hanitu supports conductance-based synapse. The synaptic current $I_{syn}$ is given by:

$$I_{syn} = wgs(V - V_{rev}),$$

where $w$ is a weighting factor, $g$(ns) is the maximum conductance of the synapse, $s$ is the gating variable and $V_{rev}$(mV) is the reversal potential for the synapse. The gating variable is described by a single exponential decay:

$$\frac{ds}{dt} = -\frac{s}{\tau} + \delta,$$

where $\tau$(ms) is the time constant and $\delta$ is a delta function which is ∞ at the time of an spike input and 0 elsewhere. All the differential equations are solved by the fourth-order Runge-Kutta method with a timestep of 0.1 ms.

| Parameters of model | Parameters in GUI | Parameters in the circuit file |
|---|---|---|
| $\boldsymbol{w}$ | Weight | Weight, Mweight |
| $\boldsymbol{g}$ | Conductance (G)<br>(In synapse part) | G |
| $\boldsymbol{V_{rev}}$ | Reversal Potential | RRevPot, SFvl, SCvl, Mvl |
| $\boldsymbol{\tau}$ | Time Constant (tau) | Tau, SFtau, SCtau, Mtau |

## 5.3. Odorant diffusion

In the virtual world, the odorant molecules propagate through the space via diffusion process. However, the Hanitu system does not solve the diffusion equation in real time. Instead, it uses the well-known finite-source solution in which the odorant concentration $\mathrm{C}$ at any time and location is given by:

$$\mathrm{C}(\Delta\mathrm{t},\mathrm{r}) = \frac{N_0}{4\pi D(\Delta t)} e^{-(r^2/4D\Delta t)}$$

where $\Delta\mathrm{t}$ is the time after the food or poison is dropped at the source location, $\mathrm{r}$ the distance from the source, $N_0$ the total amount of molecules and $D$ the diffusion coefficient. In Hanitu’s default setting $D$ is small and $\Delta t$ is large so that $\mathrm{C}(\Delta\mathrm{t},\mathrm{r})$ is nearly constant during typical virtual worm simulation periods (several seconds).

# 6. Obtaining Hanitu

The users can download the Hanitu system package and the sample configuration files of virtual world and the virtual worm circuits from http://hanitu.net.

License information:
The source code of GUI is released under GPL version 3.
The source code of Hanitu (the virtual world) is released under GPL version 3.
The Flysim neural network simulator is currently not open-sourced.

# 7. The developer team

The Hanitu System is developed by Chung-Chuan Lo laboratory (http://life.nthu.edu.tw/~lablcc/index.html) in National Tsing Hua University, Taiwan. The development team of the current version of Hanitu includes the following members:

Chung-Chuan Lo (cclo@mx.nthu.edu.tw)：Projector leader
Yu-Chi Huang (aiappleg@gmail.com)：Flysim developer
Fang-Kuei Hsieh (fangkui117@gmail.com)：Hanitu developer
Ta-Shun Su (antoniost29@gmail.com)：GUI and scripts developer

# Appendix A. Configuration files

The Hanitu system has two configuration files:

1. world_config.wcg：Virtual world configuration file. E.g. the location of worm, diffusion concentration parameters of foods. The users can set filename of circuit configuration files in this file.

2. circuit.ccg：Neural circuit configuration file of virtual worm. The users can change filenames in virtual world configuration file (.wcg).

The users can edit these two parameter files through GUI for avoiding errors.

## A.1. Virtual world configuration file (world_config.wcg)

| | |
|---|---|
| `SetWormInf` | Start of worm-related parameter setting |
| `UserID=0` | Each virtual world can host worms from multiple users. Here we set the parameters for the first user (ID=0) |
| `WormID=0` | Each user can have several worms presented in the virtual world. Here we set the first worm (ID=0) for user #0 |
| `InitialX=0`<br>`InitialY=2` | Worm’s initial location in x and y. The unit is one step (0.1mm) of the worm. |
| `Wormsize=1` | Radius of the worm. Same unit as above. |
| `Time_decay=0.1` | Constant energy level decay per second. (Initial value = 100) |
| `Step_decay=0.5` | Energy decay per step of the worm |
| `Filename=Circuit.ccg` | The circuit configuration file of the worm #0 |
| `UserID=0`<br>`WormID=1` | |
| `…` | (omitted) |
| `Filename=Circuit2.txt` | |

|  |  |
|---|---|
| `EndSetWormInf` |  |
| `SetWorld` | Now set the virtual world |
| `WorldPar` | Define the virtual world parameters |
| `Nutrient=20` | The energy level increased or decreased when a worm touches a food or a toxicant, respectively |
| `TransformA=10`<br>`TransformB=5` | Define the relation between the sensory neuron input rate and the odorant concentration (food to food sensory neuron):<br>firing rate(Hz) = TransformA × [Odor] (mM) + TransformB |
| `TransformA_FT=0`<br>`TransformB_FT=0` | (food to toxicant sensory neuron)<br>firing rate(Hz) = TransformA_FT × [Odor] (mM) + TransforB_FT |
| `TransformA_TT=0`<br>`TransformB_TT=0` | (toxicant to toxicant sensory neuron)<br>firing rate(Hz) = TransformA_TT × [Odor] (mM) + TransforB_TT |
| `TransformA_TF=0`<br>`TransformB_TF=0` | (toxicant to food sensory neuron)<br>firing rate(Hz) = TransformA_TF × [Odor] (mM) + TransforB_TF |
| `Boundary=50` | The boundary (between -50 and 50 for both x and y) of the virtual world. The unit is 0.1mm. |
| `Type=0` | Criteria for the simulation termination. Type=0, terminated when all worms die. Type=1, terminated when a worm first touches a food or a poison. |
| `Depth=0.264` | Height of the world. Used for calculating the odorant concentration. Do not alter. |
| `CountMode=1` | Do not change this parameter. It tells the program that the amount of food in each |

| | |
|---|---|
| | in each food source does not decrease when the food is eaten by the worms. |
| Fixed=0 | Fixing location of virtual worm or not. 1 for fixing and 0 for not. |
| FoodLocation | Now define where the food sources are and other related parameters. |
| FID=1 | The first food source (ID=1) |
| X=1 | The location (in 0.1 mm) |
| Y=2 | |
| Count=100 | The food count at each food source |
| Diffuse=0.002 | The diffusion constant of the food molecule$cm^2/sec$ |
| Concentration=100 | The concentration of the food molecule (mM/count) |
| Delay_time=10000 | The virtual worm simulation starts with a delay after the foods are placed in the assigned location. This parameter specified the delay time (0.1 msec) |
| FID=2 | If more food sources are need, define them (start with FID=2) before EndFoodLocation. |
| … | (omitted) |
| EndFoodLocation | End of the definition for all foods |
| MoleculeLocation | Define the parameters for toxicants |
| TID=1 | The first poison source (ID=1). The meaning of the reset parameters are the same with those for the food sources. |
| X=3 | |
| Y=2 | |
| Count=10 | |
| Diffuse=0.00005 | |
| Concentration=10 | |
| Delay_time=72000 | |
| EndMoleculeLocation | |
| EndWorldPar | |
| EndSetWorld | |

## A.2. Circuit configuration files

| | |
|---|---|
| `%meta.header=circuit_config`<br>`%meta.version=1.2`<br>`%meta.createtime=1427965147154` | Descriptions after % used by GUI. The users just ignore these. |
| `Total_neuron_number=9` | Number of neurons in the circuit. The sensory and motor neurons are not included. |
| `NeuronID=0` | Defining the first neuron (the neuron ID should start from 0) |
| `C=0.5` | Membrane capacitance (nF) |
| `G=25` | Membrane leak conductance (ns) |
| `NRevPot=-70` | Membrane reversal potential (mV) |
| `ResetPot=-55` | Reset Potential (mV) |
| `Threshold=-50` | Spike threshold (mV) |
| `Refperiod=20` | Refractory period (timestep = 0.1 ms) |
| `Spikedelay=18` | Spike delay (timestep = 0.1 ms) |
| `MembranceNoise` | Defining membrane noise (current) |
| `std=0.005` | The standard deviation of the membrane fluctuation due to the noise (mV) |
| `mean=0.1` | The mean of the fluctuation (mV) |
| `EndMembranceNoise` | End of noise definition |
| `EndNeupar` | End of the definition of somatic parameters for the neuron. |
| `ReceptorPar` | Start of the synaptic receptor definition |
| `Receptor=0` | Receptor type ID, starting from 0 |
| `Type=0` | Type of model (currently only support the type 0: one exponential decay, voltage independent) |
| `Tau=20` | Time constant of the gating variable (ms) |
| `RRevPot=0` | Channel reversal potential (mV) |
| `EndReceptor` | |

| | |
|---|---|
| `EndReceptor` | End of the synaptic receptor definition for the receptor 0 |
| `Receptor=1`<br>`Type=0`<br>`Tau=20`<br>`RRevPot=-70`<br>`EndReceptor` | Defining the second receptor type |
| `EndReceptorPar` | All receptors have been defined |
| `Targetneuron=1` | Defining a synaptic connection with a target neuron (neuron ID=1) |
| `Receptor=0` | Targeting the receptor 0 on the target neuron |
| `Weight=5` | Synaptic weight (this is just a scaling factor) |
| `G=2.5` | Synaptic conductance (nS). The synaptic strength = Weight x G |
| `EndTargetneuron` | End of the definition of the synaptic connection for the target neuron #1 |
| `Targetneuron=8`<br>`Receptor=0`<br>`Weight=9`<br>`G=2.5`<br>`EndTargetneuron` | Defining another synaptic connection with neuron #8 |
| `Endneuron` | End of the definition of the neuron #0 |
| `NeuronID=1`<br>…<br>`Endneuron` | Defining the second neuron #1<br>(omitted) |
| `Communication` | Now defining how the circuit interacts with the world |
| `Inputneuron` | Specifying input (or post-sensory) neurons which neuron receive input from sensory neurons |

| | |
|---|---|
| `NeuID=0` | Neuron 0 receiving sensory input |
| `Receptor=0` | The sensory input target the receptor 0 on neuron 0 |
| `Weight=1` | Synaptic weight for the sensory neuron input |
| `G=2.5` | Synaptic conductance (nS) |
| `Type=0` | type of sensory input (0=food, 1=toxicant) |
| `Direction=0` | Receiving sensory neuron input from the top side (0, 1, 2, 3 for top, down, left, right) |
| `NeuID=2` | Neuron 2 also receiving sensory input |
| `Receptor=0` | |
| `Weight=1` | |
| `G=2.5` | |
| `Type=0` | |
| `Direction=1` | |
| | |
| `NeuID=4` | |
| … | (omitted) |
| `Direction=2` | |
| | |
| `NeuID=6` | |
| … | (omitted) |
| `Direction=3` | |
| | |
| `EndInputneuron` | |
| | |
| `OutputNeuron` | Defining how the circuit sends output to motor neurons. Only four output (or pre-motor) neurons can be specified here. |
| `NeuID=1`<br>`NeuID=3`<br>`NeuID=5`<br>`NeuID=7` | Neuron 1, 3, 5, 7 sending output to the motor neuron on the top, down, left, right side |
| | |
| `EndOutputNeuron` | |

| | |
|---|---|
| BodyPar | Now defining sensory and motor neurons on the body walls |
| Mcm=25 | Membrane capacitance (nF) |
| Mtau=10 | Synaptic time constant (ms). This is the synapse for the connection from the output neurons to the motor neurons |
| Mweight=10 | Synaptic weight (synaptic strength = synaptic weight x synaptic conductance) |
| Msilence=20 | Neuron refractory period (0.1 ms) |
| Mvth=-50 | Spike threshold (mV) |
| Mvl=-70 | Membrane resting potential (mV) |
| Mreset=-55 | Reset Potential (mV) |
| SFcm=25 | Setting the parameters for the food sensory neurons. |
| SFtau=20 | |
| SFweight=10 | |
| SFsilence=20 | |
| SFvth=-50 | |
| SFvl=-70 | |
| SFreset=-55 | Setting the parameters for the toxicant sensory neurons |
| SCcm=25 | |
| SCtau=10 | |
| SCweight=10 | |
| SCsilence=20 | |
| SCvth=-50 | |
| SCvl=-70 | |
| SCreset=-55 | |
| EndBodyPar | |
| EndCommunication | |