Life and Evolution of Cells

Created by: Yevgeniy Gluhoy
Shaked Basa

Course: Functional Programming in Concurrent and Distributed Systems

Instructors: Dr. Yehuda Ben-Shimol and Mr. David Leon

Table of content:

Overview

Application creates a simulation of cells life and evolution under influence of the environment conditions and another cells.

Main purpose is to check our abilities to implement program, using received skills of Erlang OTP, that can deal with a big number of users and keep system stable during a simulation.

Each cell is Erlang process which interact with relevant Erlang gen server.

User interface was done using *wxWidgets*.

Technical Requirements

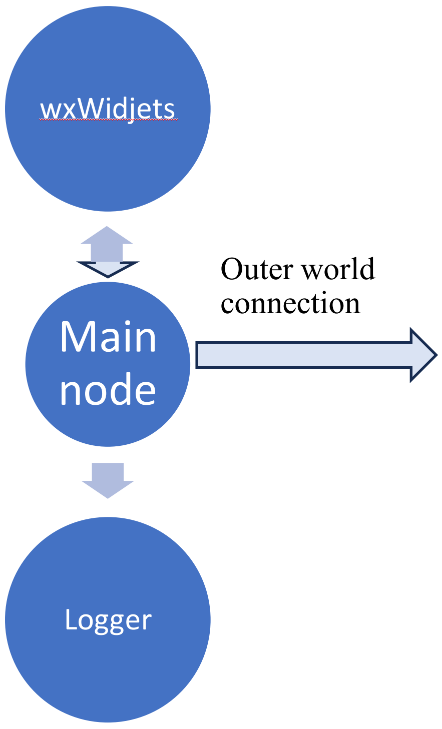
1. Apllication needs minimum 2 Erlang nodes to run simulation. One node called "Main node" and "General nodes".
2. Nodes might be distributed - different machines or can run in the same computer. In second case, each general node needs to be launched from different directory to prevent overlapping of log files.
3. Application developed on Erlang OTP 25, but also tested on Erlang OTP 24.
4. World size might be always bigger than number of cells at start of simulation. (More detailed explain could be found in "Design" chapter)
5. Full installation and requirements description is in INSTALL_GUIDE.txt on our GitHub, link can be found in last chapter.

Design

The whole system and each node have a hierarchical structure, but all nodes can communicate between them.

ETS was used as storage for database.

Main node



1. Main node implemented using Erlang gen-server, which purpose is user experience (GUI) and maintaining of system (Main data storage, recovery, start procedures).
2. It has two independent parts: GUI program and Node program.
3. Additional information about GUI will be in next chapter.
4. Node program – contains 3 Erlang processes:
    - main_logger – prints information messages to a file, used for statistic and debugging.
    - Node_monitor – used for monitoring connected general nodes. In case when one of nodes is down, informs graphic_node.
    - Graphic_node – initializes servers on all connected nodes according to parameters received from user input. Divides "world" and number of processes between connected nodes during initialization process. Starts simulation when appropriate input was received, receives, and saves data from nodes. Restarts simulation from last saved point with new partition when some node down.

GUI

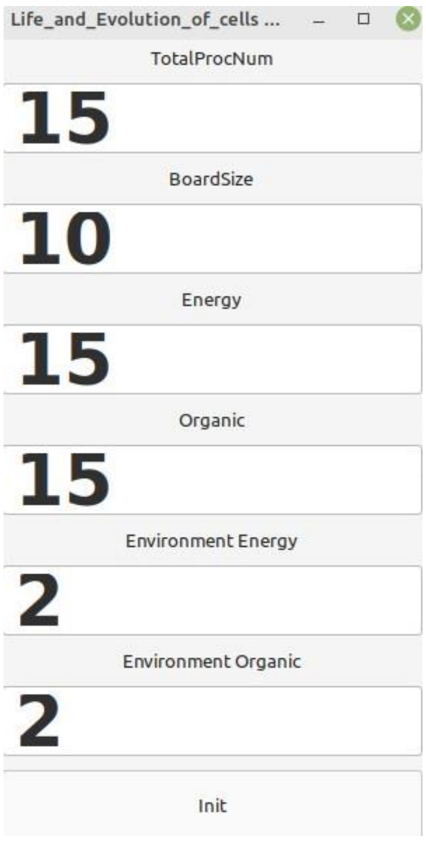


Figure 1

1. GUI implemented using *wx Widgets*.
2. Application starts from frame 1 shown in figure 1.
3. User needs to enter starting parameters: *TotalProcNum* – initial number of cells; *BoardSize* – "world" size, square with given side size; *Energy, Organic* – new cells will receive this amount of energy and organic at start; *Environment Energy, Environment Organic* – amount energy and organic in each part of the world at the start of simulation.
4. Pressing "Init" button will initialize servers, next part of user interface – frame 2 - will be opened (Figure 2).
5. Frame 2 show us current number of cells (processes) in simulation, number of connected hosts, and sun status (static in current version).
6. Pressing "Start" will start the simulation, button name will be changed to "Stop" as shown in figure 3. Pressing "Stop" will stop simulation, all servers will be closed, user interface will be changed to frame 1 and system ready for new simulation.
7. GUI use 6 Erlang processes to provide fast animation of field (shown in figure 3), even when a number of objects are big (tested on 10000 cells).

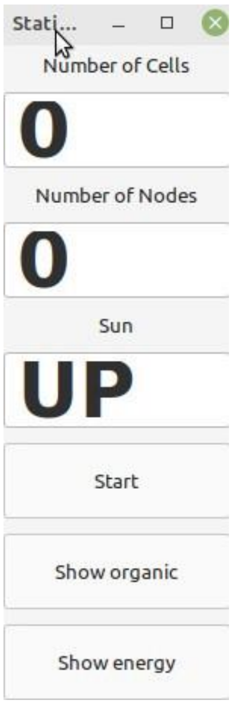


Figure 2

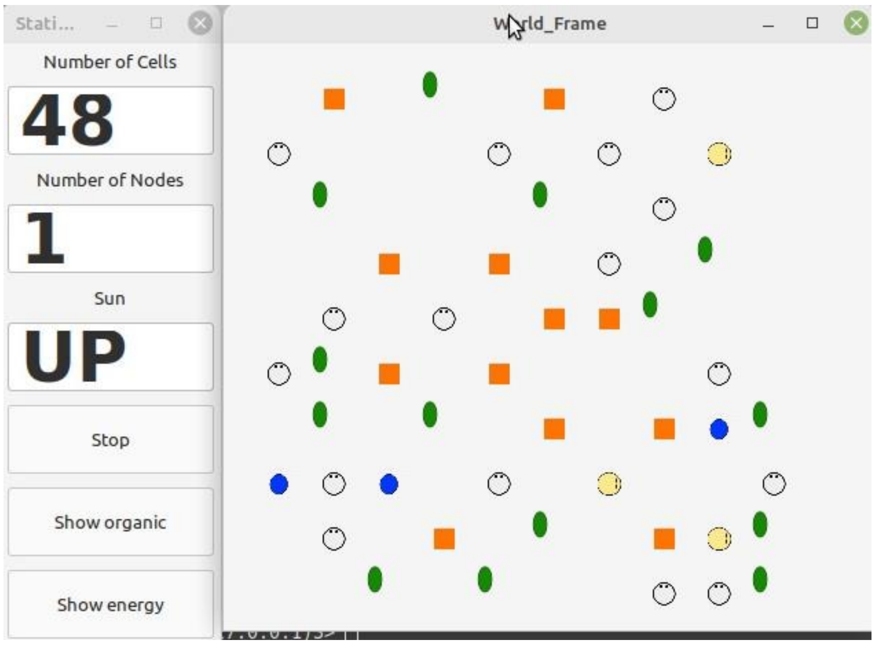
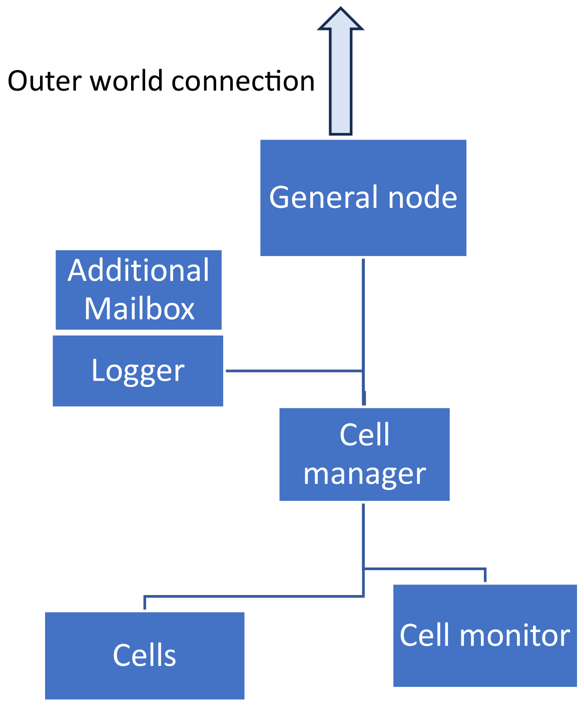


Figure 3

4

General node



1. General node is a host which runs a simulation and sends data to main node.
2. At least one general node might be online otherwise simulation will be stopped.
3. Cells are not a part of node, but they located there, they are "users" of node.
4. General node has 5 Erlang processes: cell_monitor, cell_manager, logger, mailbox and general_node.

Cell_monitor – during the simulation keeps updated list of existing cells processes, collect their exit statuses. If simulation will be stopped, cell_monitor will close all existing cell processes.

Cell_manager – implemented using Erlang gen-server. Creates cells at start and during the simulation. The only process that can change ETS table in node. When cell need to implement action, it needs to ask cell_manager, which checks if it is possible, changes data in ETS if needed and returns answer to process. In case action is move to field in world that located in another node, cell_manager sends appropriate request to general_node to create connection with another node. When cell want to move in from another node, cell_manager receives request for such action from its general_node and decides if it can be done.

Logger – simple process, which creates 2 *txt* files where it saves actions and status of cells and different parts of general node.

Mailbox – periodically sends updated database to main node and receives *ack*. If main node will not response for some time, initiates stop of simulation.

General_node – implemented using Erlang gen-server. Receives starting parameters from main node during initialization process. When main node sends message to start simulation, creates all other process in node. Stops simulation and terminates all other processes if appropriate message was received. Transports move request received from cell_manager to other nodes and from nodes to cell_manager. Can restart simulation with given initial parameters when appropriate message was received from main_node.

Cells

Cells are separate part of system. Each cell is Erlang process, most of their actions cannot be done without permission of cell_manager, but each decision made only according to current parameters of organic, energy and environment. “Life cycle” of cells is totally asynchronous relatively to other parts of system and each other, time between each iteration (decision time) is fixed, but timer starts only when answer from cell_manager was received (in some cases it will be immediately cause the answer is not needed).

Cell dies if: it has parameter TTL (Time to Live) and its value is 0; Environment energy and/or organic amount in field where cell currently located is bigger than “Critical Value” – organic/energy contamination (setup in code).

When cell dies it leaves some amount of energy and organic in field.

There are 5 types of cells in current version of application:

- ○ General cell – simulation starts only with this type of cells. Has no TTL. Possible actions: create cell (can create any other type of cell accept general), move (moving to another position, move always to near field – diagonal possible too), transform to another cell (can transform to any other type of cell), do nothing (evolution is weird process), eat (action can be found in code, but currently cell_manager always answers no for such requests cause ability not implemented in “upper layers”).
- 🟡 Seed cell – sleeping cell, has no TTL, almost not use energy. With some probability can transform to general cell in current iteration. Represents reproduction mechanism of evolution process.
- 🟢 Leaf cell – receives energy from photosynthesis. Has TTL, not use energy when sun parameter is “UP”. (In current version parameter has value “UP” always)
- 🔵 Antena cell – can absorb energy from environment, survives in places with energy contamination. Has TTL.
- 🟧 Root cell - can absorb organic from environment and transforms it into energy, can survive in places with organic contamination.

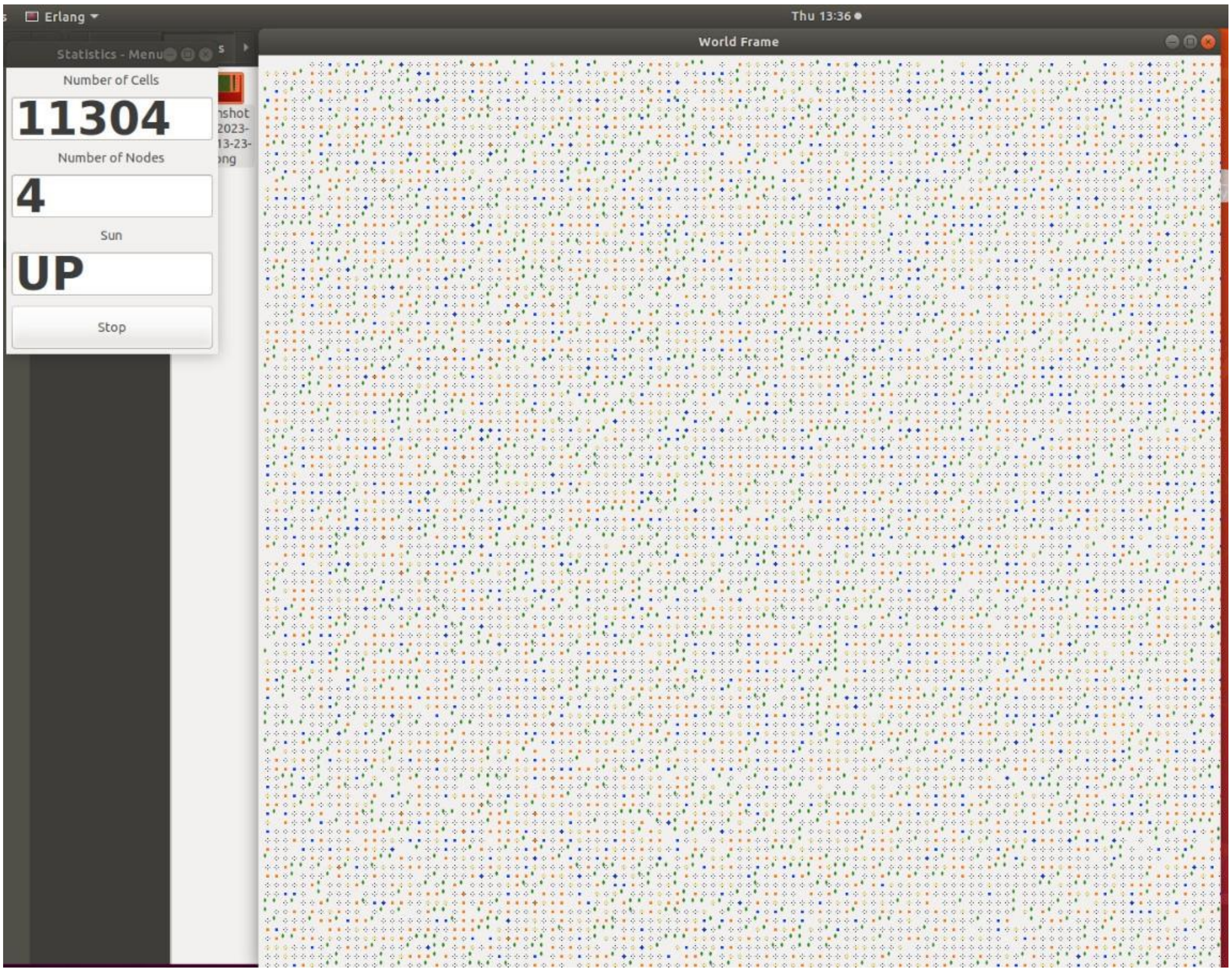
Simulation

Simulation has no fixed duration time. It will end only if all cells die or will be stopped if no available hosts left.

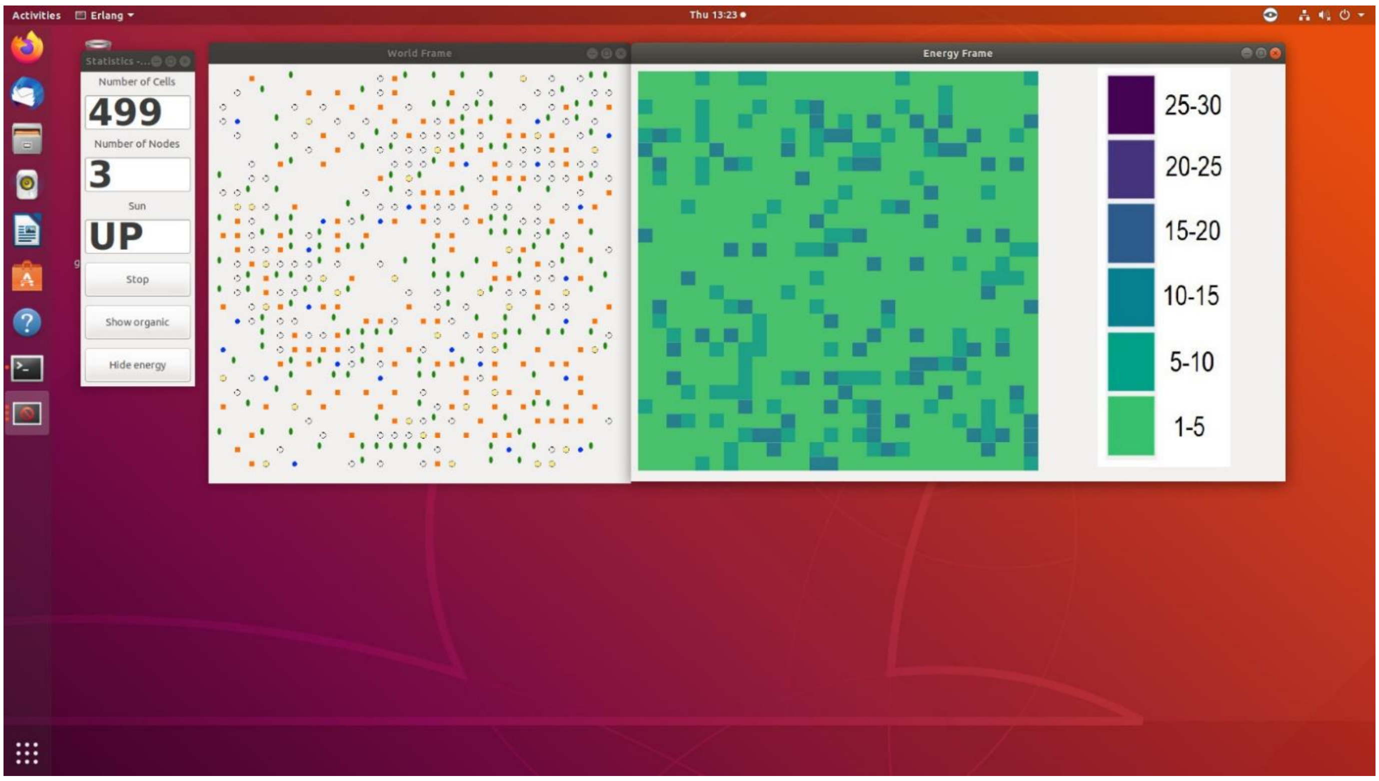
Simulation depends on many parameters: values of organic/energy contamination, iteration time, initial values of environment and cells energy/organic. In addition, cells to make decision in new round, calculate new weights for each action according to environment and status (TTL, Energy amount...) and make random uniform choice.

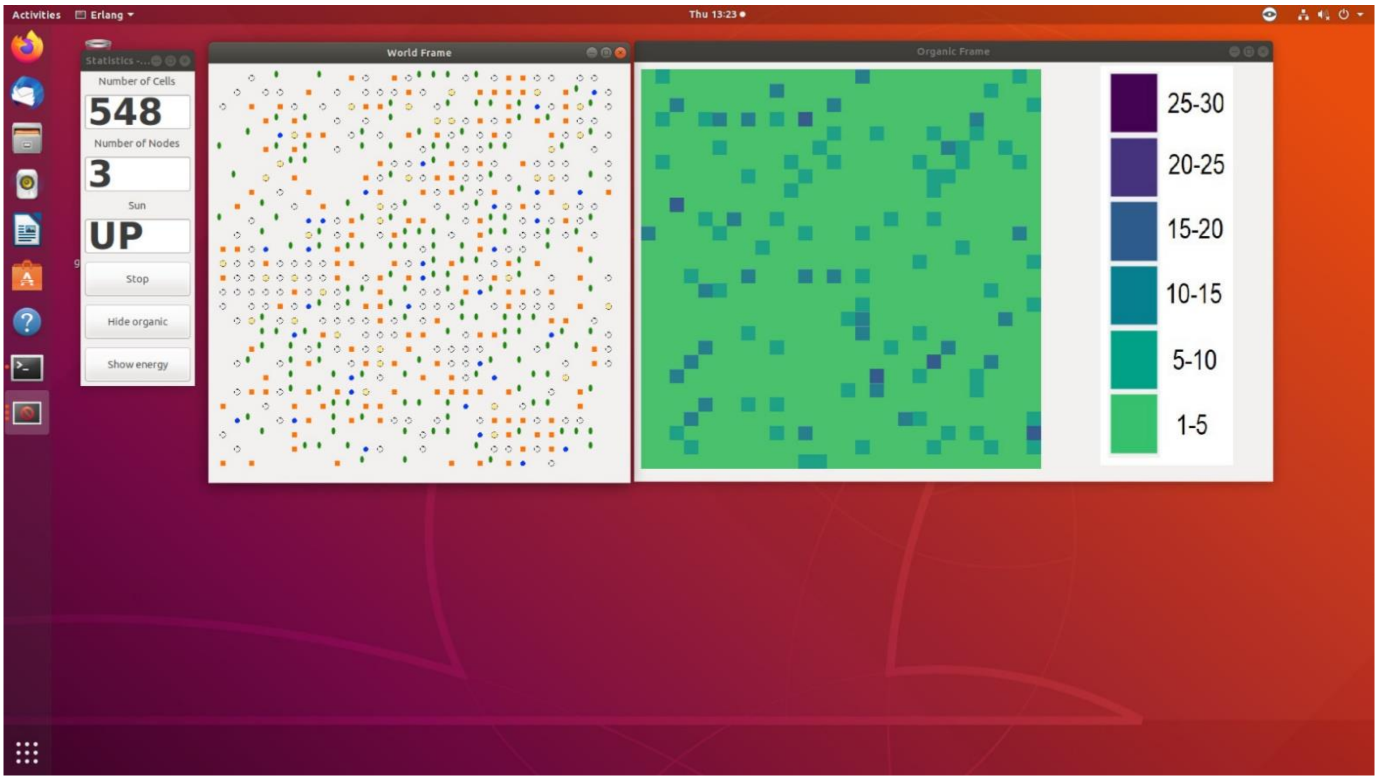
Application has 3 different version of GUI, for each one was done different tests. There is full explanation how to use them in our GitHub.

1. Standard version provides all user interface described in previous chapters. It handles up to 10 000 processes with world size 108x108 (108x108 is biggest resolution available in this version of GUI).

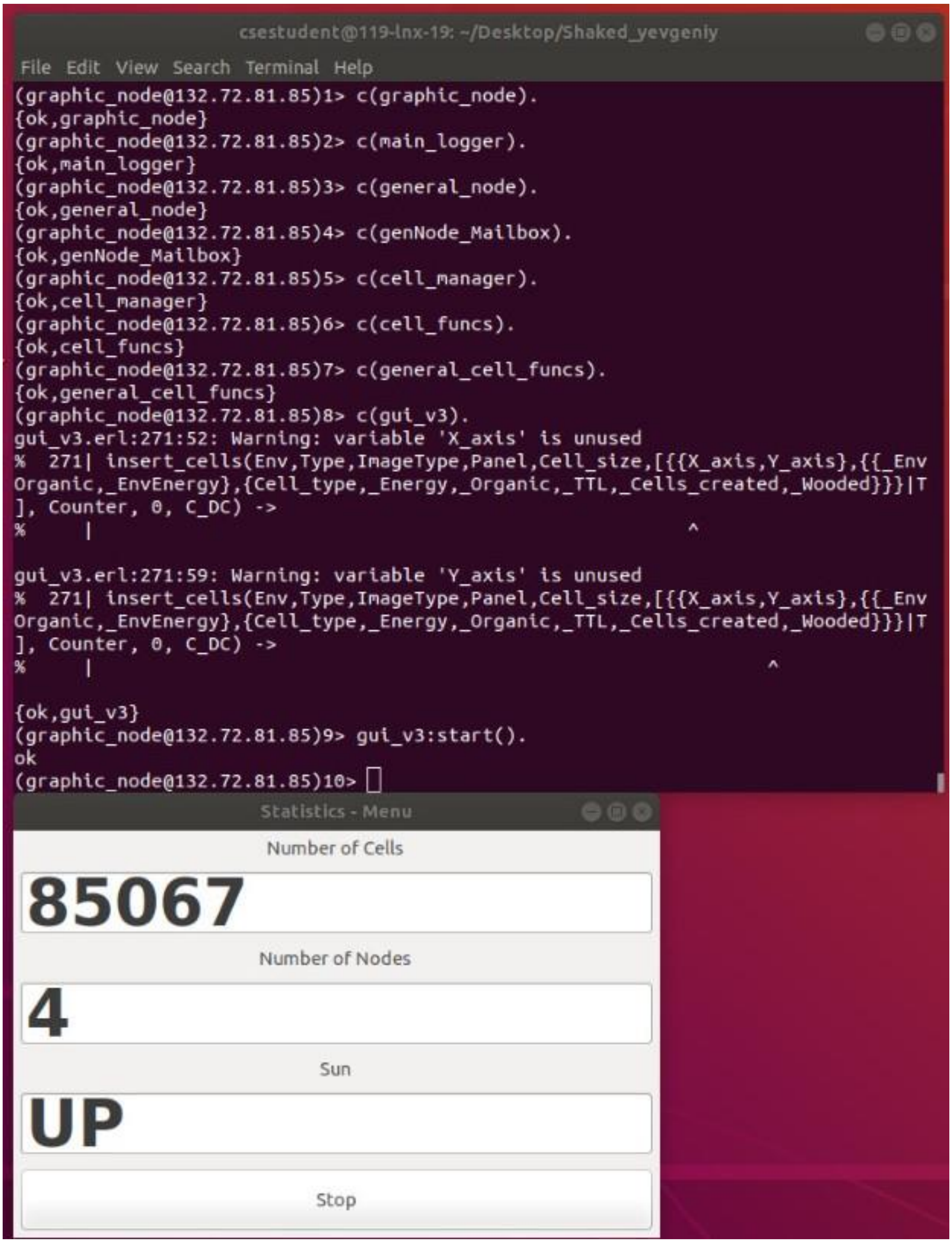


2. Heatmap version has full functionality of standard version and in addition can show frames of current organic/energy distribution in “world”. Its disadvantage is performance, handles a smaller number of objects at screen. This version was tested with up to 500 processes at start on field 30x30.





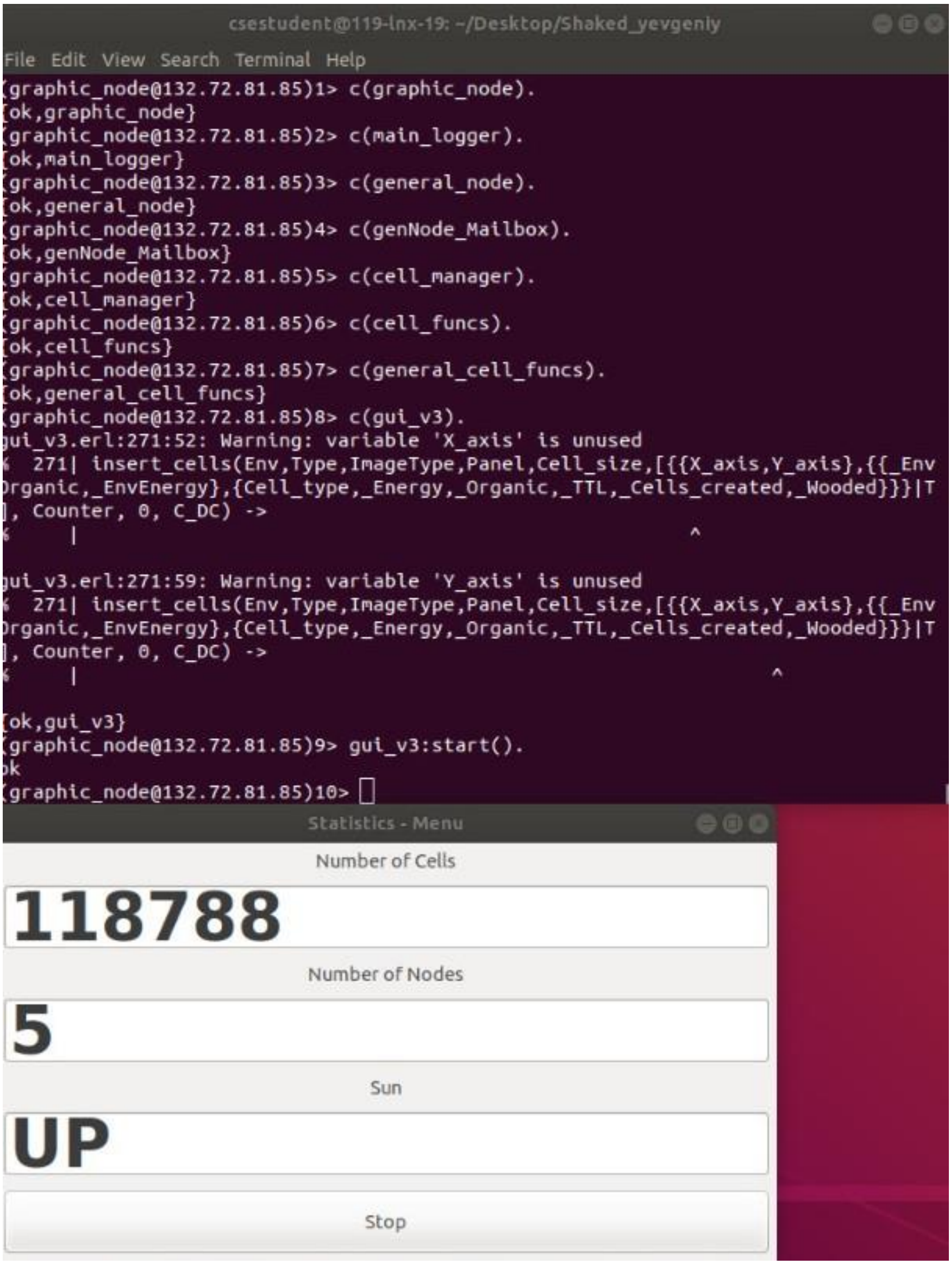
3. No GUI version was built for “stress” test of application. This version has no world frame, only statistic panels with buttons to show current number of processes and available hosts. Was tested with 50 000 process and 300x300 world size initially on 4 nodes; 90 000 processes and 350x350 world size initially on 5 nodes.


csestudent@119-lnx-19: ~/Desktop/Shaked_yevgeniy
File Edit View Search Terminal Help

```
(graphic_node@132.72.81.85)1> c(graphic_node).
{ok,graphic_node}
(graphic_node@132.72.81.85)2> c(main_logger).
{ok,main_logger}
(graphic_node@132.72.81.85)3> c(general_node).
{ok,general_node}
(graphic_node@132.72.81.85)4> c(genNode_Mailbox).
{ok,genNode_Mailbox}
(graphic_node@132.72.81.85)5> c(cell_manager).
{ok,cell_manager}
(graphic_node@132.72.81.85)6> c(cell_funcs).
{ok,cell_funcs}
(graphic_node@132.72.81.85)7> c(general_cell_funcs).
{ok,general_cell_funcs}
(graphic_node@132.72.81.85)8> c(gui_v3).
gui_v3.erl:271:52: Warning: variable 'X_axis' is unused
%  271| insert_cells(Env,Type,ImageType,Panel,Cell_size,[{{X_axis,Y_axis},{{_Env
Organic,_EnvEnergy},{Cell_type,_Energy,_Organic,_TTL,_Cells_created,_Wooded}}}|T
], Counter, 0, C_DC) ->
%     |                                                      ^

gui_v3.erl:271:59: Warning: variable 'Y_axis' is unused
%  271| insert_cells(Env,Type,ImageType,Panel,Cell_size,[{{X_axis,Y_axis},{{_Env
Organic,_EnvEnergy},{Cell_type,_Energy,_Organic,_TTL,_Cells_created,_Wooded}}}|T
], Counter, 0, C_DC) ->
%     |                                                             ^

{ok,gui_v3}
(graphic_node@132.72.81.85)9> gui_v3:start().
ok
(graphic_node@132.72.81.85)10> 
```



csestudent@119-lnx-19: ~/Desktop/Shaked_yevgeniy
File Edit View Search Terminal Help

```
(graphic_node@132.72.81.85)1> c(graphic_node).
{ok,graphic_node}
(graphic_node@132.72.81.85)2> c(main_logger).
{ok,main_logger}
(graphic_node@132.72.81.85)3> c(general_node).
{ok,general_node}
(graphic_node@132.72.81.85)4> c(genNode_Mailbox).
{ok,genNode_Mailbox}
(graphic_node@132.72.81.85)5> c(cell_manager).
{ok,cell_manager}
(graphic_node@132.72.81.85)6> c(cell_funcs).
{ok,cell_funcs}
(graphic_node@132.72.81.85)7> c(general_cell_funcs).
{ok,general_cell_funcs}
(graphic_node@132.72.81.85)8> c(gui_v3).
gui_v3.erl:271:52: Warning: variable 'X_axis' is unused
%  271| insert_cells(Env,Type,ImageType,Panel,Cell_size,[{{X_axis,Y_axis},{{_Env
Organic,_EnvEnergy},{Cell_type,_Energy,_Organic,_TTL,_Cells_created,_Wooded}}}|T
], Counter, 0, C_DC) ->
%     |                                                    ^

gui_v3.erl:271:59: Warning: variable 'Y_axis' is unused
%  271| insert_cells(Env,Type,ImageType,Panel,Cell_size,[{{X_axis,Y_axis},{{_Env
Organic,_EnvEnergy},{Cell_type,_Energy,_Organic,_TTL,_Cells_created,_Wooded}}}|T
], Counter, 0, C_DC) ->
%     |                                                           ^

{ok,gui_v3}
(graphic_node@132.72.81.85)9> gui_v3:start().
ok
(graphic_node@132.72.81.85)10> 
```

Links

Short video with running application:

https://youtu.be/PmqbEnC1Po0

Project on GitHub:

https://github.com/gekaizum/Functional_programming_project