

OWorm manual

version 1.2

1. Introduction.....	1
2. Installation.....	2
3. Setting parameters	
3.1 Matrix parameters	2
3.2 Set up matrices	3
3.3 Cycle parameters	3
3.4 Worm parameters	4
3.5 Dynamic	5
3.6 Auto parameters	5
3.7 Positions	7
3.8 Identity	7
3.9 Teach	8
3.10 Odds'n ends.....	8
3.11 Error warning!	9
4. Operation.....	9
5. Sample test runs	
5.1 Manual operation, basic setup	
5.1.1 Absolute minimal settings	12
5.1.2 Maximal settings on some platform	12
5.2 Manual operation, additional features	13
5.3 Auto run operation	13
6. Monitoring	14
7. After the prototype.....	15
8. Bug report	15

1. Introduction

OWorm (derived from the associated book's title "On the origin of Mind" and Worm) represents a prototype artificial mind driving a graphic rendition of an artificial worm. Its fundamental dynamic consists of the creation, maintenance or destruction of clusters of fitness peaks within the context of fractal borders through forming affinity relationships with other clusters which in subsequent cycles become more or sometimes less confirmed.

The program is designed to be a test bench for an OtoomCM application and allows parameter settings to observe the behaviour of this kind of artificial mind when connected to an effective output.

For the technical details behind the basic system see the paper "How the mind works: the principle dynamics in a bio- and non-bio version" available on the website www.otoom.net > Downloads. For the summary details relating to the worm operation refer to OWorm on the website. Also available there under downloads is a comprehensive set of test results. This manual deals with the user interface only.

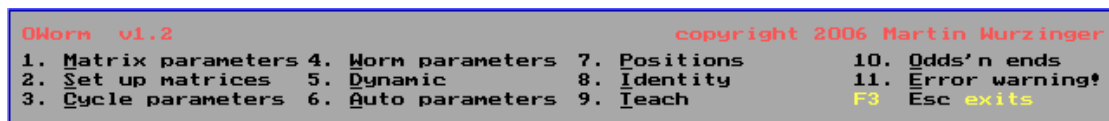
2. Installation

OWorm is a DOS-based program for the Windows platform, written in C/C++ using some graphics (this makes it easy to move the code into a larger-scale distributive environment). Click on Oworm12.exe and the program starts in a DOS window.

No changes are made to the hard drive, the operating system, or anywhere else.

3. Setting parameters

The menu section:



Select a menu by pressing the underlined letter on the keyboard.

F3 and Esc exits the program and/or a menu and either one is operational when highlighted.

If no matrix has been defined, first **Matrix parameters** must be selected, then **Set up matrices**, then **Cycle parameters**, otherwise error messages will result.

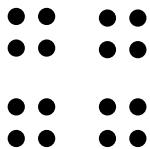
3.1 Matrix parameters

Press **M** on the keyboard.

Input window for **Matrix parameters**:



The system's dynamic operates within a matrix where each individual entry is another matrix. The matrix entries are referred to as nodes, and the entries of a node are called elements.



The above example represents a matrix with four nodes and four elements per node. Or, a matrix with two node rows and two node columns, and two element rows and two element columns per node. Such a configuration is referred to as a $2 \times 2 \times 2$ matrix.

WARNING: Due to DOS memory limitations under the Windows 98 operating system the maximum safe values are those shown in the screen shot above (*high end of elem. number range* excepted). Anything higher may crash the program (there is no technical limit to the values in terms of the program itself—but see **7. After the prototype**).

Enter the values for *node matrix rows* etc by pressing Tab to proceed left-right, top-down; for the opposite direction use Shift-Tab. Press Enter to exit each field and store the value.

Notes:

node matrix rows to *elem matrix cols*: number of rows and columns for the matrix nodes and the elements per node.

connect'y/node %: the number of nodes every single node is connected to expressed as a percentage of the total number of nodes in the matrix. Example: a value of 10% with 100 nodes in the matrix means that each node is connected to 10 other nodes.

KB: the memory used in kilobytes and calculated automatically when the respective values have been entered. The value is affected by the preceding parameters only. HINT: note the value at a maximum safe setting so that next time different combinations can be tried safely.

high end of elem. number range: the highest integer which will be used for the node elements (the minimum is 16). The higher this number, the more expressive the output will be. The number does not affect the memory usage.

3.2 Set up matrices

Press **S** on the keyboard to set up the matrix according to the parameters entered before. The memory indicator on the bottom of the screen shows the memory used (red bar) out of the total made available by the operating system, and the amount of memory left in KB as a number (green box):



Ongoing date and time values are displayed on the left and right side respectively.

3.3 Cycle parameters

Press **C**. Entering these numbers follows the same Tab system as for the matrix parameters.

A screenshot of a terminal window showing a form for entering cycle parameters. The form has a blue border and contains four input fields with labels to their left. The labels are 'input regions:', 'output regions:', 'n. disp. freq.', and 'block factor :'. The values entered in the fields are 5, 5, 0, and 1 respectively. The label 'conn. depth' is also present but has no input field next to it.

input regions:	5	output regions:	5
		n. disp. freq.:	0
block factor :	1	conn. depth :	10

Notes:

input regions: the number of sections within the input nodes.

output regions: the number of sections within the output nodes.

n. disp. freq.: update frequency of the display on the screen showing the state of affinity relationships for each node as the system goes through its cycles. Example: a value of ten means that the display will be updated every 10th cycle. HINT: calculating this display uses by far the most resources (but does not affect the inner matrix). If not required at all, set to 0.

block factor: the number of times a depth traversal is repeated.

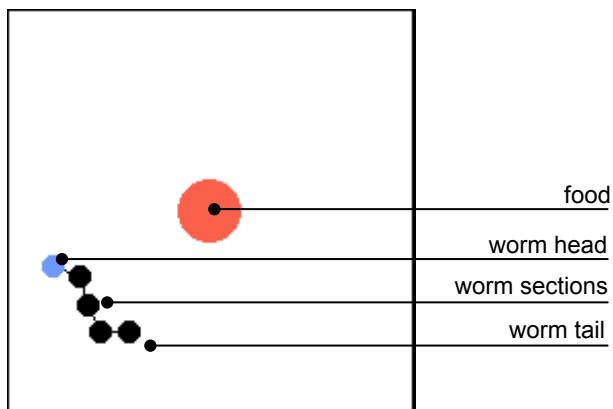
conn. depth: the number of depth levels to be traversed across the virtual trees in the matrix.

3.4 Worm parameters

Press **W**. These are the values that are subject to ongoing processing in the matrix. For other configurations of the worm see **3.10 Odds'n ends**.

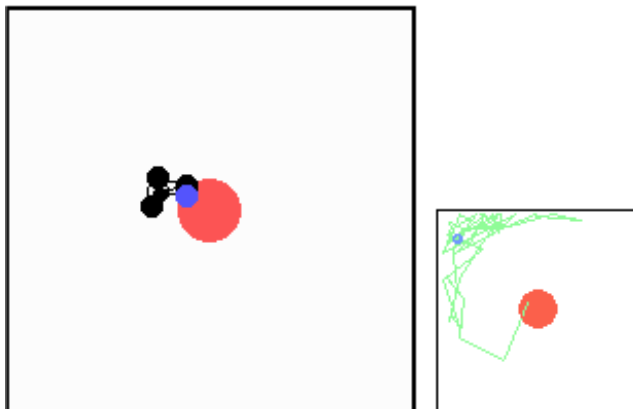
direction:	<input type="text" value="10"/>	dir. change	:	<input type="text" value="20"/>
approval :	<input type="text" value="0"/>	add. dir. change:	:	<input type="text" value="30"/>
wipe matrix after food found				: <input type="text" value="NO"/>

Screen capture of the cage during the worm's movement:



The worm moves by starting on one end and placing the following sections according to the direction value (relative to the previous orientation), as well as applying another change and an additional direction change to these values. The last segment drawn is now the anchor point on the surface and becomes the starting point for the next sequence; thus the direction changes move from head to tail and from tail to head alternatively.

The food cycle is completed when the head has reached the food area, such as below with the paths taken during the cycles alongside:



Notes:

direction: initial direction in degrees when drawing the worm either from the head or tail.

dir. change: value in degrees applied to *direction*.

approval: number of times the processing is repeated when the distance to the food has decreased (this value is added to the *block factor* and displayed on the screen).

add. dir. change: value in degrees applied to *direction change*.

wipe matrix after food found: when YES the elements in all the nodes will be reset to their initial values when the worm has reached the food and the food cycles continue. If NO the cycling will continue with the current phase states in the matrix.

3.5 Dynamic

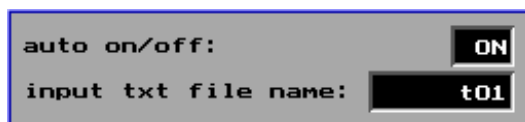
Press **D** to start the system.

Using this command to start and/or recommence the cycles will completely reset all the screen displays. To continue after a Pause (Ctrl-Z) with the current matrix content and screen values preserved, use the Spacebar.

WARNING: The program allows for input at any time from the keyboard, and that includes the auto run mode. Whether activating Pause and similar switches causes the keyboard value from the buffer to enter the program's input stream depends on the operating system. When formal testing is required it is advisable to leave these switches alone since the system's internal phase states could be affected.

3.6 Auto parameters

Press **A**. Used for test runs, when various cycle parameters and other system behaviour need to be evaluated.



Tab into *auto on/off* to toggle from OFF to ON and vice versa.

Tab into the next field to type the name of the input parameter text file (maximum 8 characters, no file extension).

When auto is ON the program writes various bitmap files at the completion of every cycle as well as the completion of the food cycle/s. The file names of these files are automatically generated and reserve the last 4 characters for numbers from 0000 to 9999, with every file consecutively numbered. The first part of the name is made up of the parameter text file name, shortened if necessary to accommodate the four count digits.

Example of a parameter input text file, with user-defined values in bold:

```
oworm
t01.txt
food_cycle_number:
1
teach_cycle_number:
1
```

```
element_writing:
1
cage_writing:
1
screen_writing:
1
```

Notes:

oworm: file type identifier.

t01.txt: file name, must be the same as the actual file name.

food_cycle_number: the number of times the food must be reached by the worm before the program stops regardless of how many matrix cycles it takes. Must be at least '1'.

teach_cycle_number: the number of times a teaching cycle is executed. In teach mode it must be at least '1'.

element_writing: for writing the element values to a text file. '0' off, '1' on. This file will contain all the element values of all the nodes organised cycle by cycle. Do not edit. NOTE: depending on the size of the matrix and/or the number of elements per node as well as the number of cycles performed, this file can get quite large.

cage_writing: when set to '1' the area in which the worm moves about is written to a bitmap file after every matrix cycle.

screen_writing: when set to '1' after the food has been reached the entire screen area except for the menu section is written to a bitmap file.

WARNING: Do not alter the format of the parameter input file. Use '0' to cancel a parameter.

Example of the file naming system:

input parameter text file name given by the user: *t01.txt*

the generated files are-

element text file name: *T01-e.txt*

cage file names: *T01c0001.bmp*, *T01c0002.bmp*, *T01c0003.bmp*...

screen file names: *T01s0001.bmp*, *T01s0002.bmp*, *T01s0003.bmp*...

worm movement file names: *T01m0001.bmp*, *T01m0002.bmp*, *T01m0003.bmp*...

cycle statistics file name: *T01-c.txt*

food cycle statistics file name: *T01-f.txt*

Example from an element text file:

```
...
cycle 12
node 1
5511 4375 4083 2891 2474 1777 1537 739 31
node 2
4860 4869 3559 3103 2284 1819 1228 288 29
node 3
5000 4375 3750 3125 2500 1875 1250 625 0
node 4
5213 4261 3911 2856 2404 1898 1342 734 25
node 5
4904 4327 3697 3016 2284 2153 1317 429 16
node 6
5211 4664 4065 3414 2711 1957 1152 948 21 ...
```

Example from a cycle statistics file:

Cycle statistics for OWorm

cycle no.	food cyc. no.	distance	block factor
1	1	62	1
2	1	50	1
3	1	50	1
4	1	37	1
5	1	37	1
6	1	64	1
7	1	64	1
8	1	87	1
9	1	87	1
10	1	72	1
11	1	72	1...

Example of a food cycle statistics file:

Food cycle statistics for OWorm

food cyc. no.	cycle no.	mean distance	aggregate time
1	426	94	00:00:25

3.7 Positions

Press **P**. There are four default starting positions for the worm at the beginning of each food cycle. The x/y values are in pixels at the program's screen resolution and are measured from the top left corner of the cage. The maximum value for x and y is '194'.

pos1 x:	<input type="text" value="20"/>	pos1 y:	<input type="text" value="30"/>
pos2 x:	<input type="text" value="30"/>	pos2 y:	<input type="text" value="160"/>
pos3 x:	<input type="text" value="165"/>	pos3 y:	<input type="text" value="45"/>
pos4 x:	<input type="text" value="155"/>	pos4 y:	<input type="text" value="160"/>
cycle through positions : <input type="text" value="NO"/>			

Notes:

pos1 x / pos1 y to *pos4 x / pos4 y*: respective starting positions for the worm within the cage.

cycle through positions: if YES the entered starting positions are cycled through automatically in sequence after a food cycle has been completed. When in auto run mode this has to be coordinated with *food_cycle_number* in the parameter input text file.

3.8 Identity

Press **I**. Rather than starting the worm movements with the initial integers inserted when the rows and columns were created, up to ten values can be used to seed the matrix, from 1 to several times. The more often the process is repeated, the more matrix nodes will have been affected.

NOTE: these numbers are not simply spread across the node elements; they are input which is processed by the system.

1:	3	2:	46
3:	97	4:	158
5:	262	6:	299
7:	317	8:	386
9:	402	10:	495
block factor:		0	

Notes:





1 to 10: identity input values.

block factor: number of times each value is processed by the system. If '0' the values have no effect.

3.9 Teach

Press **T**. An algorithm guides the worm from the active starting position to the food, using the three direction parameters to move it across the surface. During the movement the parameters serve as input to the system where they are processed, but the resultant output is discarded. Once the teach mode has been completed the worm is reset to the starting position and the matrix takes over, now containing phase states derived from the previous movements.

Four modes can be selected, where a higher number represents a less direct path to the food.

mode 1:		mode 2:	
mode 3:		mode 4:	
select teach mode:		0	

Notes:

mode 1 etc: degree of directness towards the food.

select teach mode: enter the mode number. '0' switches teach mode off.

3.10 Odds'n ends

Press **O**. Several parameter values can be changed in order to affect the output. They are not significant for evaluating the system's nature as a mind simulation, but can be interesting to observe the influence they have over the behaviour of the worm.

low value	:	1	food radius	:	15
worm sections:		5	worm link length:		15
turn limit of worm sections			:	130	
reverse approval factor			:	OFF	

Notes:

low value: minimum value for the number range used to fill the node elements during the creation of the matrix (the maximum is selected at *high end of elem. number range* in **Matrix parameters**). NOTE: this entry has been deactivated in the current version to keep the default at '1'.

food radius: radius of the food disc in pixels at the program's resolution.

worm sections: number of sections in the worm, minimum set to '5'.

worm link length: length of the line between each section in pixels at the program's resolution.

turn limit of worm sections: maximum number of degrees at which each section of the worm can be turned in relation to the previous orientation, regardless of the system's output. It simulates a physical constraint in a mechanical device driven by some outside command.

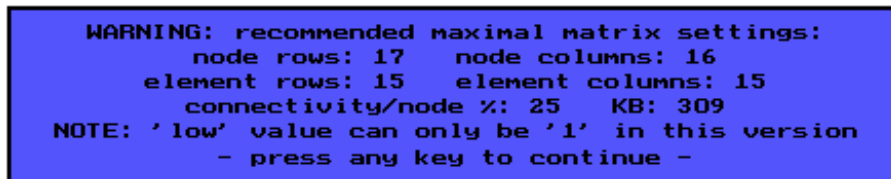
reverse approval factor: when ON the *approval* setting in **Worm parameters** is reversed, ie the input is processed a selected number of times minus the value in the *approval* field when the worm's distance to the food has decreased. For example, if *block factor* = 4, *approval factor* = 3, and *reverse* is ON, when the worm's distance to the food has decreased the effective *block factor* is now 1; otherwise the effective *block factor* would be 7.

3.11 Error warning!

Press **E**. Due to memory limitations imposed by the Windows operating system regarding DOS programs and any hardware in general the matrix cannot be sized up without limits.

Exceeding the limits will crash the program as soon as **Set up matrices** is pressed.

Under Win98 the limits are as below.



```
WARNING: recommended maximal matrix settings:
node rows: 17   node columns: 16
element rows: 15 element columns: 15
connectivity/node %: 25   KB: 309
NOTE: 'low' value can only be '1' in this version
- press any key to continue -
```

4. Operation

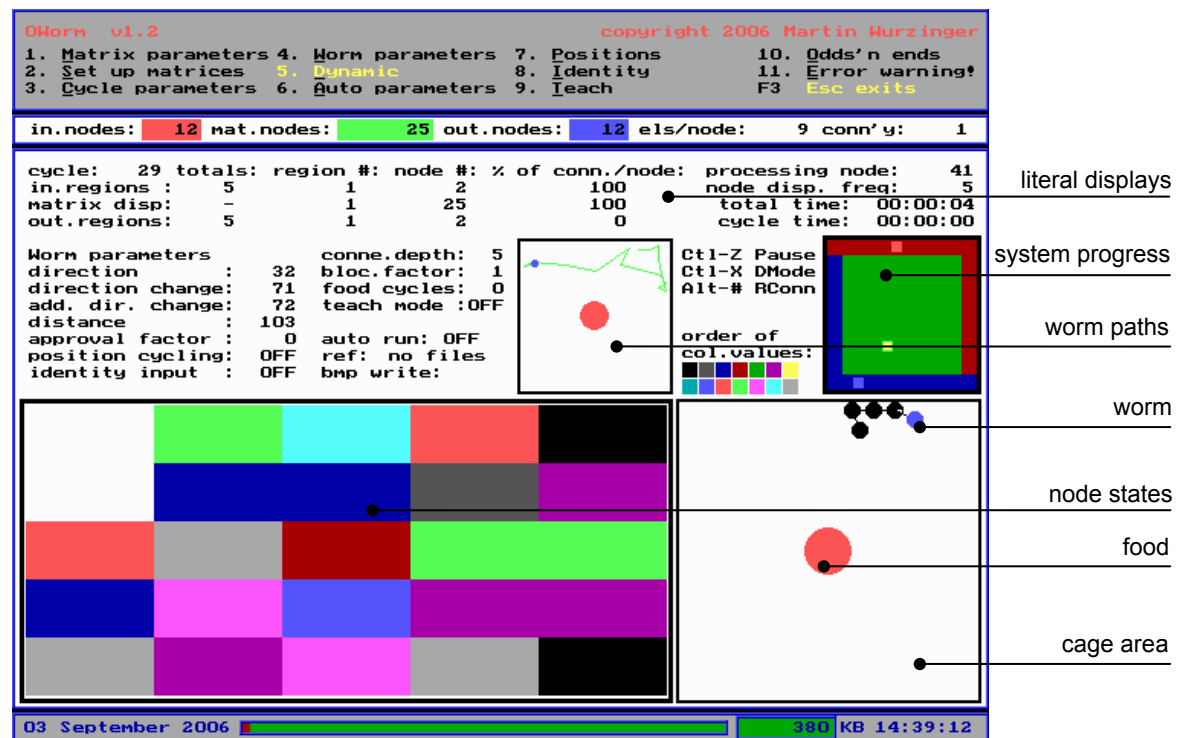
Operation is started by pressing **D**. The system will cycle continuously, regardless of the presence of input. During cycling, pressing any key on the keyboard which is not designated for a particular purpose stops the current cycle and starts it once again with the new input value (ie, the integer value of that key). This also applies to auto run. To stop the cycles completely, press Esc (sometimes repeatedly) to exit the menu.

To pause the cycles press Ctrl-Z, to restart press the Spacebar.

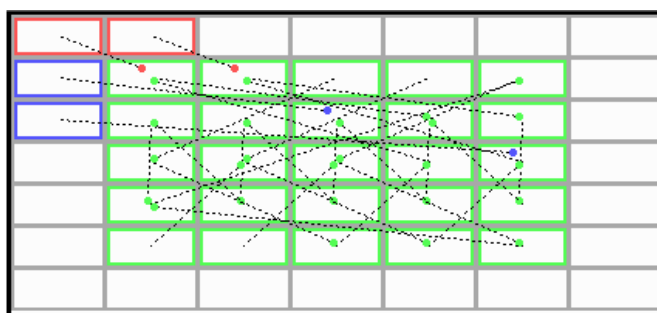
WARNING: The program allows for input at any time from the keyboard, and that includes the auto run mode. Whether activating Pause or DMode (or switching windows) etc causes the keyboard value from the buffer to enter the program's input stream depends on the operating system. When formal testing is required it is advisable to leave these switches alone since the system's internal phase states could be affected.

HINT: If viewing the affinity relationships and layout plan is not necessary, the operation will be speeded up significantly by setting *n. disp. freq* (under **Cycle parameters**) to '0'.

With the affinity relationships displayed the screen looks like this:



If the update frequency of the display has been set to '0' when setting the *Cycle parameters*, no node states will be shown. If set to '1' or higher, Ctr-X will toggle between a display of affinity relationships for every node (shown above under *node states* with every colour-coded rectangle representing a node in the inner matrix) and the layout plan of the matrix, showing the active input region (red), the active output region (blue), the processing nodes of the matrix (green), and the connections between all the nodes. In the example below there are 5 in- and 5 output regions and 25 inner matrix nodes; connections to an inner matrix node from an input node are indicated by a red dot, those to an output node by a blue dot, and links within the inner matrix nodes are shown in green.



Notes:

in.nodes: number of input nodes

mat.nodes: number of inner matrix nodes

out.nodes: number of output nodes

els/node: number of elements per node

conn'y: number of nodes connected to each node

cycle: number of cycle just completed

totals: number of total in- and output regions

region #: current in- and output region being processed (incl matrix which is always '1')

node #: current node of in-, output region, and matrix (if display is 'on') being processed

% of conn./node: percentage of virtual node tree in in-, output region, and matrix processed

processing node: node in the inner matrix being processed (= parent node of the virtual tree)

node disp. freq: when *n. disp. freq* selected the time/cycles at which affinities are calculated

total time: time counter for one test run

cycle time: time counter for one cycle

direction: initial direction in degrees when drawing the worm either from the head or tail

direction change: value in degrees applied to *direction*

add. dir. change: value in degrees applied to *direction change*

distance: direct line between the worm's head and the centre of the food disc in current pixels

approval factor: additional cycle/s applied to inner matrix nodes if *distance* has decreased

position cycling: if 'on' the next food cycle automatically starts at the next position

identity input: a set of values to seed the inner matrix nodes

conne.depth: number of levels for the virtual trees at each inner matrix node

bloc.factor: additional cycle/s applied to inner matrix nodes at every cycle

food cycles: number of times the food has been reached when on auto run

teach mode: when 'on' an algorithm guides the worm to the food, not the system

auto run: when 'on' the parameter input text file must be used

ref: designator of the test run, basis for all the files generated by the program

bmp write: when a screen area is written to a bitmap file a 'w' blinks on and off

Ctl-Z, Ctl-X: toggle switches for Pause, Display Mode if display is 'on'

Alt-#: selector for in- and output region by number (= #) if display is 'on'

order of col.values: for node display, low to high affinity from left to right, top to bottom

Bottom section: date, memory used of available total (bar), memory left (integer), time

5. Sample test runs

5.1 Manual operation, basic setup

The following settings provide for manual operation without any special factors affecting the system's output. Characters from the keyboard can be used as additional input while the system is cycling.

5.1.1 Absolute minimal settings

Press **M** for **Matrix parameters** and press Tab and Enter to get the following default values:

node matrix rows:	<input type="text" value="7"/>	cols:	<input type="text" value="7"/>
elem matrix rows:	<input type="text" value="2"/>	cols:	<input type="text" value="2"/>
connect'y/node %:	<input type="text" value="5"/>	KB :	<input type="text" value="4"/>
high end of elem. number range:	<input type="text" value="16"/>		

Press **S** for **Set up matrices**.

Press **C** for **Cycle parameters** and press Tab and Enter to get the following default values:

input regions:	<input type="text" value="5"/>	output regions:	<input type="text" value="5"/>
		n. disp. freq.:	<input type="text" value="0"/>
block factor :	<input type="text" value="1"/>	conn. depth :	<input type="text" value="5"/>

Press **D** for **Dynamic**.

NOTE: When a test run is completed a tone sounds and the Stop message appears:

```
STOP - food found!  
- press any key to continue -
```

Press Esc/F3 to exit the menu and/or the program any time or when the food has been reached.

5.1.2 Maximal settings on some platform

Press **Matrix parameters** and type in the following values:

node matrix rows:	<input type="text" value="17"/>	cols:	<input type="text" value="16"/>
elem matrix rows:	<input type="text" value="15"/>	cols:	<input type="text" value="15"/>
connect'y/node %:	<input type="text" value="25"/>	KB :	<input type="text" value="309"/>
high end of elem. number range:	<input type="text" value="32767"/>		

Press **Set up matrices**.

Press **Cycle parameters** and type in the following values:

input regions:	5	output regions:	5
		n. disp. freq.:	32767
block factor :	32767	conn. depth :	15

NOTE: *in-* and *output regions* are set to '5' for this version because of the number of in- and output values needed for the worm, and *conn. depth* cannot exceed '15' because of the matrix size.

Press **Dynamic**.

Press Esc/F3 to exit the menu and/or the program any time or when the food has been reached.

5.2 Manual operation, additional features

Go through the steps shown in 5.1 and select any settings within the range identified by the values shown in 5.1.1 and 5.1.2.

Select any menu (**Worm parameters**, **Positions**, **Identity**, **Teach**, **Odds'n ends**) or a combination of them and type in the desired settings.

Press **Dynamic**.

Press Esc/F3 to exit the menu and/or the program any time or when the food has been reached.

GENERAL NOTE: After a test run has been completed or has been stopped by pressing the Esc key, a different matrix size and different cycle parameters can be chosen by selecting **Matrix parameters**, **Set up matrices**, and **Cycle parameters** in that order and entering the desired values. If the matrix size is different the existing matrix will be collapsed and redeployed under the new settings and initialised.

5.3 Auto run operation

Prepare the parameter text file. Select an appropriate name to reflect the test run/s and edit the name in the file itself. Ensure that the following file entries synchronise with the menu entries later on:

```
food_cycle_number - Positions > cycle through positions
teach_cycle_number - Teach > select teach mode
```

The chosen cycle numbers must allow for multiple cycles and/or the mode to be valid in the first place.

NOTE: Editing the name in the file itself may seem tedious, but prevents errors in the test regimen (especially when there are dozens of files to be considered).

Select the menu **Matrix parameters** and enter the desired values, press **Set up matrices**, select **Cycle parameters** and enter the desired values.

Select any of the other menus (**Worm parameters**, **Positions**, **Identity**, **Teach**, **Odds'n ends**) and enter the desired values.

Press **Auto parameters** and tab into *auto on/off* to set to ON. Tab into *input txt file name* and type the name of the parameter text file (excluding file extension).

Press **Dynamic**.

Press Esc/F3 to exit the menu and/or the program any time or when the food has been reached.

The program will halt when the worm has reached the food in the food cycle, or in the case of position cycling all the food cycles have been completed. The generated files can be found in the program's directory/folder.

NOTE: If a formal set of test runs is planned where the results must be compared against some common standard, do not press any keys while the system is cycling. The content of the keyboard buffer could enter the system and produce stray effects. The nature of the system makes it impossible to trace any stray input.

NOTE: Depending on the processor speed of the CPU some settings can result in test runs spanning several days (especially with large matrix sizes). Exact times are impossible to predict.

NOTE: When setting *teach_cycle_number* to a value greater than '1' the teach algorithm will move the worm on different paths on subsequent teach cycles. With certain starting positions, number of worm sections, particular worm link lengths, and certain turn limits of worm sections the worm may end up cycling the food disc without ever making contact with its head. In this case press 'Esc' to reset the current vectors used for adjusting its direction parameters. The current teach cycle will be repeated.

GENERAL WARNING: Due to the highly interdependent nature of the processes within the matrix, the intermittent results during a test run depend on the individual timing between the processes (of which there can be millions). This is influenced by factors such as hard disk access speed, bus speed, CPU speed, and their respective caches, even the subdirectory level the program resides in! Therefore, if the purpose of several test runs is to compare the results with each other it is vitally important to conduct the tests in similar directory sublevels. Otherwise there is no guarantee the results (such as the number of cycles before the food is reached) can be meaningfully compared with each other.

6. Monitoring

The generated bitmaps and text files allow the system's performance to be monitored in a formal manner across a series of test runs.

Setting the *element writing* flag to '1' in the parameter text file (provided *auto on/off* is set to ON) will generate a text file containing the complete set of element values for the entire duration of the test run, ordered by cycles and nodes. This file must not be edited.

To extract the value of a certain element of a particular node across the cycles, use the *pickel* program (*pickel.exe*). For example, if the element text file is *t1-e.txt* (generated from the input parameter text file name *t1.txt*), the output file is meant to be *p1.txt*, the chosen node at each cycle is the 45th node in the matrix (left-right, top-bottom), and the chosen element at this node is the 16th element (left-right, top-bottom), then the command line syntax would be

```
pickel t1-e.txt p1.txt 45 16
```

After processing, the generated file p1.txt contains the integer values of the respective element arranged in one row (suitable for insertion into a spreadsheet). If the test run had 30 cycles there will be 30 integers.

7. After the prototype

The program needs a hardware environment which allows the matrix to be scaled up to extremely high levels (eg, the human brain contains ca 100G neurons, but depending on memory even that value can be exceeded). For this the code does not need to be altered.

To provide a reasonable real-time response and processing, the format needs to be changed from the current linear processing to a distributed one. The required code changes would be minimal as the code is already highly modular and the operations across the matrix are all of the same type.

There is no theoretical limit to the amount and types of in- and output, provided an appropriate interface is built so that the system receives the input as an integer, and the output translates from an integer to its respective format. The code only requires a copy/paste per additional in- or output.

Currently the integer values relating to the matrices are stored as type 'int' to conserve memory. This limits the range of any number between -32768 and +32767 (equal to 2 bytes). Therefore the integers resulting from processing the algorithm that go beyond that range will be represented mathematically incorrect, although that does not invalidate the behaviour in terms of stable, periodic, or strange attractors (the integer values are not important, but their mutual relationships are). Large-scale versions where higher correct values are needed require a type 'long int' memory allocation. 'Long int' storage (equal to 4 bytes) allows for numbers between -2,147,483,648 and +2,147,483,647.

At the scaled-up level, testing would involve going through parameter combinations in a systematic manner and evaluating them in terms of such overall criteria as expectability, consistency, and independence from any outside control/feedback, as well as the abstractive reach of content and the degrees of affinity with related and not so related input.

8. Bug report

To my knowledge there are no problems with the menus as such. However, due to the special nature of the program unexpected behaviour can occur under certain conditions, mostly due to the specific environment provided by the operating system (I have found a couple and corrected them).

Please report any problems to martinw[AT]otoom.net.