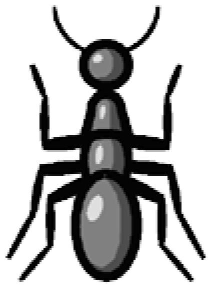


Cellular Automata: Langton's Ant Model

Written By:

Lance Baker

Research Report

Introduction

The following Research Report document will:

- Provide an in-depth explanation, stating the purpose of cellular automaton; with the surrounding history illuminating how the conception originated.

- Explain the *Langton's ant* model; emphasizing the original set of rules, and clarifying the behaviour generated. Including detailed evaluations of the initial chaotic behaviour; with an explanation of the emergent formed.

Overview

- Langton's ant is a two dimensional Turing Machine that is designed to simulate cellular automaton, or cellular automata. It was first developed by Christopher Langton in 1986, with the objective to demonstrate his studies based on the premise that 'simply defined rules lead to complex resulting behaviour'.

- The model uses a simple set of rules that produce extremely complex behavior pattern. This emergent complex behaviour has been baffling mathematicians, as despite of the initial configuration; the behaviour witnessed, is deemed to be chaotic, though the ant amazingly retransforms the erratic mess into a state of order – building an unbounded highway structure, over a series of self-replicating motions.

Cellular Automaton History

Cellular automation, or cellular automata, is a discrete model used in many fields such as computability theory, mathematics, physics and others. Cellular Automaton was studied in the 1950's as a possible model for biological systems and has since been comprehensively studied by S. Wolfram in the 1980's.

Cellular Automata was first introduced throughout the work devised by *John von Neumann,* in who was trying to develop an abstract model for organic self-reproduction (based on the conception of membrane mitosis), he theorized in creating a model for which the behaviour could be artificially replicated. His research commenced in trying to base 'the model' in three-dimensional aspects, and attempted to construct an example using a mechanical construction set. Later, he contemplated (after complications emerged with the electronic circuitry) that two-dimensions should be sufficient. Midway throughout 1952, *Neumann* had constructed 2D cellular automaton encompassing 29 possible states for each cell; with a complicated definition of rules, in which was capable of self-reproduction. He later stated that complex system designs would be unavoidable in order to emulate complex 'life-like' behaviour.

Throughout, the 1950s computing platforms emerged that could have easily performed cellular automata simulations (but were pre-occupied with other studies), and within 1960 a couple of simulations relating to cellular automata was finally undertaken. During the 1960s studies underwent into the aspects of using simulated geometrical objects (researched primarily by *Stanislaw Ulam*); in order to develop recursive pattern behaviour. It was noted throughout his research, that simple growth rules generated a complicated behaviour (with also a mention to the relevance to biology).

In 1961, *Edward Fredkin* developed a self-replicating 2D automaton (but was primarily focused on discovering ways to emulate physics-like characteristics), and later during the decade *John Conway* in 1968 commenced experimentations following the research conducted by Neumann, Ulam, and Fredkin. *Conway* devised a simple set of rules exhibiting complex erratic behaviour, in which he called ‘The Game of Life’, and was widely popularized due to a published article in the ‘Scientific American’ magazine.

Throughout the decade of 1970, development in projects continued, and offshoots into new branches of practical application were studied. The popularity boosted by ‘The Game of Life’ started to decrease, and interest in the field was on a steady declination; which ultimately gave reasoning for *Stephen Wolfram* to conduct a research paper (with keen expectation to rebirth the field into a new vision). In 1981 his research paper was first published, in which identified the key generic aspects of creating self-replicating patterns, remerging the notion that simple constructed designs are capable of emerging complex behaviour (therefore refuting *Neumann’s* statement).

The interest within the field increased from the publication, sparking a new era of research; which as a direct result inspired *Christopher Langton* in 1984 to formulate the theory behind “Langton’s Loop” using simply defined conditions with the goal of creating a self-replicating iteration based on artificial genetic information. In 1986, his research was continued and basing on the same concepts discovered in the late 60s (by *Conway)* he constructed another version of automaton; that was simple in design (using two key rules), yet emerged complicated self-replicating behaviour.

Langton's Ant Model

Christopher Langton, a well renowned computer scientist, and one of the founding fathers whom conceived the term 'Artificial Life'; devised the 'Virtual Ant' model based on direct inspiration, and studies underwent by *Stephen Wolfram*.

The 'Virtual Ant' *(*later named after its creator as *"Langton's Ant")* is originally derived on the notion first proposed by the 'Cellular Automata' model, in which is a study theorizing that biological cell membranes can be simulated based on a defined set of rules that govern how the cells converse, and interact. This therefore, implies that the basic fundamentals of life itself could be artificially emulated in order to simulate biological processes for study, even being adapted to computerized systems for intelligent application designs, to self-replicating systems, and essentially marking a paradigm for cellular-robotics.

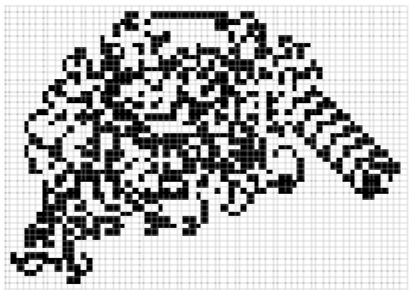
The model is based on the concepts of organic life itself, and the cells are indeed representations of cellular membranes; in which are portrayed as a two-dimensional blocks within a computerized grid. The cellular interactions are triggered on intervals, and once initiated; the starting point is commenced, with the next position amongst the grid being determined in correspondence to the neighbouring cellular states. The model uses a defined set of rules that inevitably produces (after a number of iterations) a complex emergent pattern. The shear notion of the emergent appearing out of a chaotic state has been baffling mathematicians; as there is no-real easy explanation for its existence.

Moreira, Gajardo, & Goles (2001, p. 47) stated that the "lack of an answer for such an 'easy' question about a simple deterministic system has even been cited as an example of the possible explanatory limitations of a 'theory of everything'".

The grid is composed of cellular states; where each cell can be in a finite number of options; for example as an “on” & “off”, in-whom are represented by colouring within the grid cells. As the *Langton’s Ant* progresses, each cell met mutates its state according to a pre-defined rule set (based on the state of the neighbouring cells). The grid once commenced can be visually observed as the cells advance, and overtime self-replicating patterns emerge.

Ant Behaviour

The patterns generated on the grid; solely depends on the ruling definitions setup within the system. The most well-known pattern is based on the simplest ruling founded, in which creates the ‘highway’ emergent behaviour. Once initiated, the leading point appears to encompass a circular type motion of erratic randomness behaviour for a period of 10, 647 steps. Afterwards, the leading point is iterated for a constant pattern of 104 steps in which seems to build a recursive high-way like structure leading outwards diagonally; which continues to build this path-like structure for an indefinite amount of time (proven via the *Cohen-Kung* Theorem).



The movement of the original *Langton’s Ant* is based on the following rule-set:

1. If the ant is on a black square, it will turn east and move forward one unit.
2. If the ant is on a white square, it will turn west and move forward one unit.
3. When the ant leaves a square, the colour is inverted.

The *chaotic behaviour* depicted from the ant’s movement, may seem erratic, but the pathway isn’t random. The initial pattern generated (before the emergent) will be identical to each cycle; with the only exception being the orientation of the figure, which depends on the originating directional bearing (north, south, east, or west) of the ant; resulting in the pattern being vertically or horizontally rotated.

Rule Strings

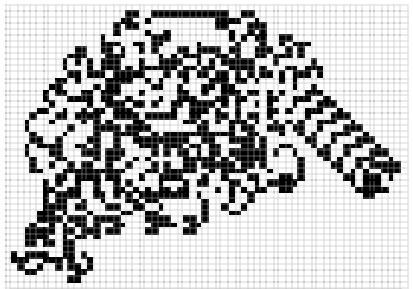
Greg Turk, and James Propp (throughout their research) devised a further extension to the studies conceded from the original model – by theorising that additional cellular-sates could be defined by a string of rules, leading to the conception of 'rule strings'; allowing for definition sets to be constructed based on binary representation.

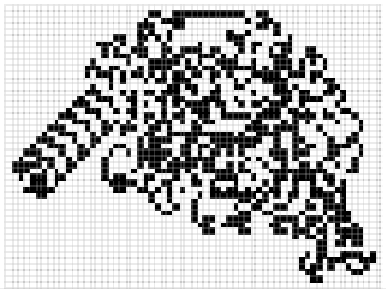
The original *Langton's ant* model was formulated on the definition set of {1, 0} cellular states, with each state being represented by a different colouring. The rule string itself, could be envisioned as an array of values; functioning on the following generalized design conceptions:

- After each state encounter, the point must advance to the next cellular position, with the direction corresponding to the current cellular state value. If the cellular-state encountered is [1], then the ant must traverse to the eastern bearing, otherwise it will navigate to the west.

- After leaving an encountered cell, the state must be flipped to the next state in order; which is calculated based on the (current value + 1) modulus (size of (rule string) -1).

Rule strings consisting of all 1s or all 0s are insignificant; since the ant will simply turn in the same bearing (east or west respectively) each time. Likewise, the mirroring of all the bits in a rule; will result in an inverted image being generated:

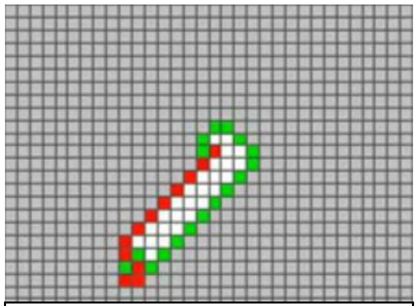

Rule string: {1, 0}


Rule string: {0, 1}

Emergent Behaviour

The path-like emergent behaviour is believed to have been accidentally discovered by *Christopher Langton*; whilst observing possible variants of inputted rules based on his studies for creating a simple cellular automaton program.

The explanation for the existence was then later indepthly researched by Mathematician James 'Jim' Propp; who conducted a series of experimentations using the *rule string* generic model. His studies found evidence stating that under specific circumstances where the cellular-states of {1, 0} in subsequent ordering eventually led to an emergent path-like pattern design; in which he coined as the 'highway' emergent.
A 'highway' pattern example (as shown in the right image) is built from adding a 'third' cellular state into the equation; which alternates the path immensely, by just building the self-replicating path-like structure after 80 steps; with each successive piece taking 18 steps to build (instead of the original 104).


Rule string: {1, **1, 0**}

Rule String	*Highway*
1 0	*yes (104)*
1 0 0	*no*
1 0 1	*no*
1 1 0	*yes (18)*
1 0 0 0	*no*
1 0 0 1	*no*
1 0 1 0	*yes (104)*
1 0 1 1	*no*
1 1 0 0	*no*
1 1 0 1	*yes (388)*
1 1 1 0	*yes (52)*

A newly devised multiple definition set (greater than two) containing {1, 0} however, is not guaranteed to transpire into a self-replicating pattern; as there will be some circumstances where such a pattern will only emerge.
The data table to the right is a summary of the observations (up until the length of four), and whilst viewing the possible variants; it can be noted that rule strings pertaining {1, 1, 0} and {1, 1, 1, 0} have the same common pattern. In which he then recognized that the {1, 0} rule can be preceded by any given number of 1s; which as a result will always construct a 'highway' pattern design.

Cohen-Kong Theorem

The Cohen-Kong theorem is a mathematical concept, stating that the current path created by the model is time-reversible; meaning, that the current cellular-state uniquely determines both the past, as well as the future. The theorem also provides evidence supporting the notion that the trajectory of the emergent is of an unbounded nature; repeating the same emergent ‘building-blocks’ indefinitely.

Conclusion

- The original Langton’s ant ruling definition set (consisting of two cellular-states) seems to initially behave in a chaotic erratic fashion for a period of *10, 647* steps, until eventually emerging to a self-replicating *104* step pattern called a highway.

- The trajectory of the emergent pattern is unbounded; proven via the *Cohen-Kong* theorem.

- Rule strings allow for the inclusion of more than two cellular-states, and follow a generic design – potentially allowing for an unlimited amount of states to be incorporated in the model.

- Emergent patterns are transpired from the rule string values {1, 0}, and can be exhibited in multi-cellular state instances.

In concluding, the ingredients for the *Langton's Ant* model, are simplistic enough in design; to allow for ease-of experimentation, by just fiddling around with the possible state variants, and observing the pattern generated. In the instances where a self-replicating pattern has been produced; it should be noted that the ant appears to consist of a sophisticated course of action – comprising itself of a series of reproducing steps; formulating a type-of ongoing construction. For an unacquainted observer the ant would appear to have been originally devised to achieve this purpose; where-in reality the ant was never produced to have this specific behaviour in mind. Therefore, the impression delivered from the model is that complex behaviour doesn’t necessary require a design, and could indeed be created without an originating purpose in question.

Vision of the Solution

Vision Statement

The vision is to create a Cellular Automaton based on the concepts proposed by Christopher Langton. The system would be a Python based application, which encompasses a grid-like layout; with a starting leading point. The system should include the ability to change the definition of the cellular states (being able to add the different states via another form), once the rules definitions have been entered – the system should start based on a thread, which iterates the steps; checking the conditions added each time it cycles.

The results from each cycle will be exported to a database table, and the system should provide the ability to view the results in a line graph. The 'history' section of the system should display the past 'ant cycle' results in a data-grid, and once the record has been double clicked – the system then opens another form displaying the graph. The graphing system should be an outside plug-in, and it has been requested from the sponsor to include the ability to 'Zoom in'.

List of Features

1) The grid will need to developed, which has enough cells to appropriately display the developed patterns. The grid should ideally be dynamic, allowing the ant to roam around an infinite amount of cells – panning outwards as the ant builds a larger structure. It is though, unknown whether this could be accomplished; so the display-area may just be a fixed range of pixels.

If the developed grid is a fixed size; there could be a potential issue surrounding the fact the there is a boundary. So therefore, the system could simply 'stop' the thread once the ant has simply ran-out of room. The grid will need to be large enough though, to compensate for the space required to include the erratic randomness behaviour – before the emergent appears.

2) The Langton’s ant system, should allow for the ability to dynamically add additional cellular states (defaulting to black and white). The cellular states will be represented by colours; with the colours being chosen from a type of colour-picker. Once the colour has been selected, the user should then choose the direction the ant will travel once the state has been encountered. The possible directions should be selectable in the form of a combo-box; with the available bearing being listed as options.

The cellular state following the previous ruling definition will be the colour that the state will be inverted to once the cell has been exited, with the last definition looping back to the first one. Therefore, the rules entered would allow for the system to dynamically classify each definition – not being limited to the two standard rulings.

3) The system should log the results of each cycle in a database table, storing the cell position (based on the x and y axis), with also the date/ time of the cycle. The system should also include the ability for the past cycles to be revised, with the data being displayed in a line-graph (using the x, and y axis).

a) The history, should be selectable from the main window – and be displayed as a separate child form; containing the data-grid of results, accompanied by a dynamic graph (located above the data-grid in a separate panel).
b) The data-grid should be using pagination (initially showing a set number of records; being divided across a series of pages).
c) The columns on the data-grid should be sortable and also searchable within a specific date-range.
d) The first record within the data-grid will be selected by default, and once a record has been newly selected; the graph will be refreshed showing the corresponding data.

4) The system should have the ability to pause/ resume the iteration, with also the option to terminate the cycle. The speed for which the thread is operated-on, should be scalable – allowing the user to increase, and decrease the speed suiting to their personal preference. The use-interface should also display the speed for which the thread is running at (in milliseconds), with also an incremented 'Time-step' counter; in which will be a grand total of iterations.

Project Scope

Features that <u>will be</u> included

1) The final system should be able to demonstrate the "Langton's Ant" ideologically for any given duration; providing adequate amounts of time for the system to generate the desired recursive patterns.

2) Once the 'Ant cycle' has been terminated by the user, the pattern should have an option of being saved (in an image format); which will dynamically scale out to an altitude including the entire generated pattern.

3) The history for the cycles will be stored in a database table, and will be viewable within a separate form; showing the results in a line-graph (based on cell position/ time step).

4) Ideally, the grid will be dynamic in size. However, if this objective is unobtainable we will use a grid that is in a fixed-range, and the 'Ant Cycle' will be automatically terminated once the boundary has been met (though, the grid-size should still be large enough to show the eventuating patterns).

5) The system will be created based on dynamic interchangeable set-of rules, which will be defined in the systems options. The default rules will be the ones originally devised by Christopher Langton.

6) The system will be composed of a series of external libraries, and it will not be 'reinventing the wheel' with some aspects. A suitable graph that is compatible with Python will need to be found, and incorporated into the project.

Features that will not be included

1) The system will be using a locally stored DBMS, and will not be internet-based in this release. Support for an external connection though; will be considered when implementing the data layer.

2) The system will only be compiled as a desk-top application, and will not be compatible will applet technologies (therefore, it will not be solely created to publish on a website). However, given the chosen technologies – slight modification could warrant in this ability to be accomplished for a further release.

3) The system will be only created based on the concepts proposed by Christopher Langton's research, and will only consist of a single originating leading point. The application won't consider this prospect throughout the design, so therefore it won't be deemed applicable for further implementation on a later phase.

Interface Designs

Menu Tree Structure

File

- ➔ New Run
- ➔ Save Pattern
- ➔ Exit

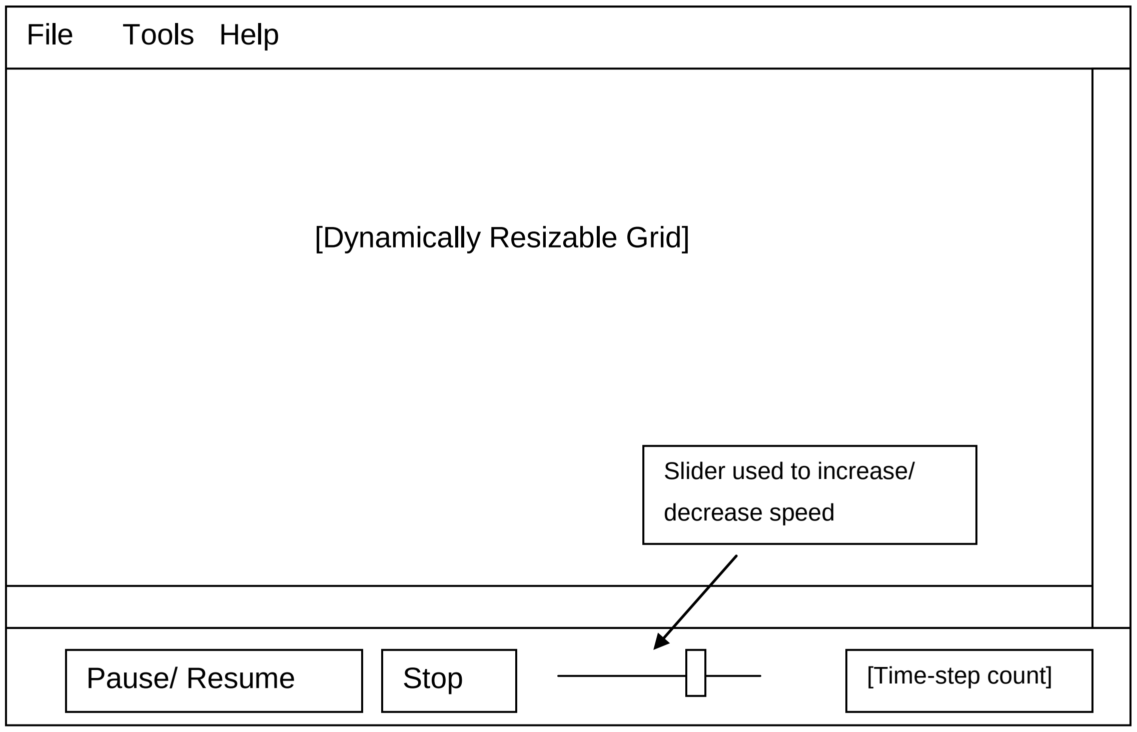
Tools

- ➔ Options
    - →Pause/ Resume
    - →Stop
    - →Zoom
        - → Zoom-in
        - →Zoom-out
    - → Rules
- ➔ History

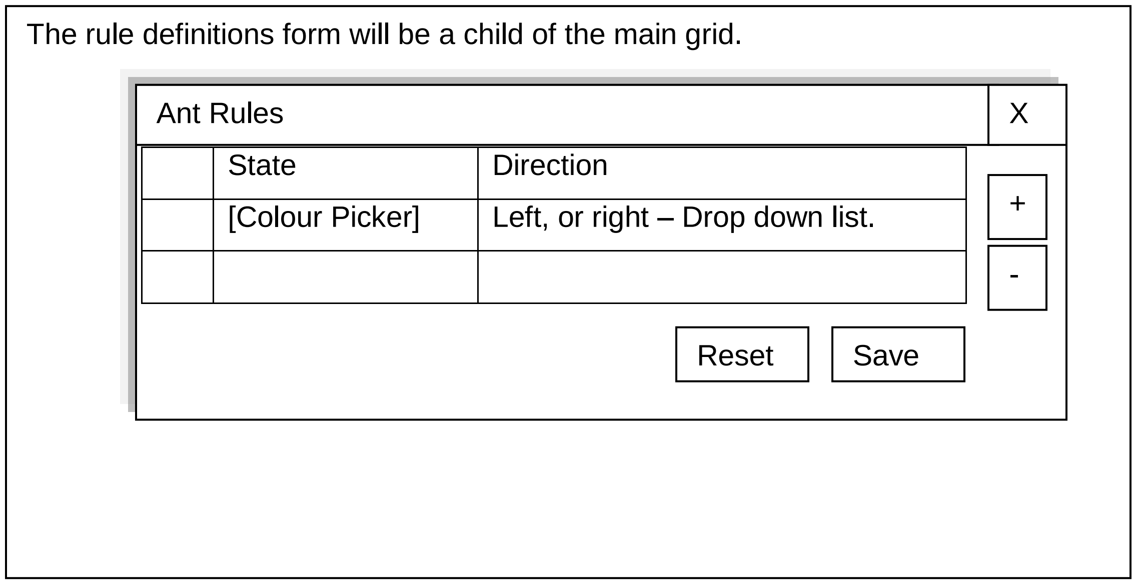
Help

- ➔ About Box

Main Form


File
Tools
Help
[Dynamically Resizable Grid]
Slider used to increase/ decrease speed
Pause/ Resume
Stop
[Time-step count]

Rules Form

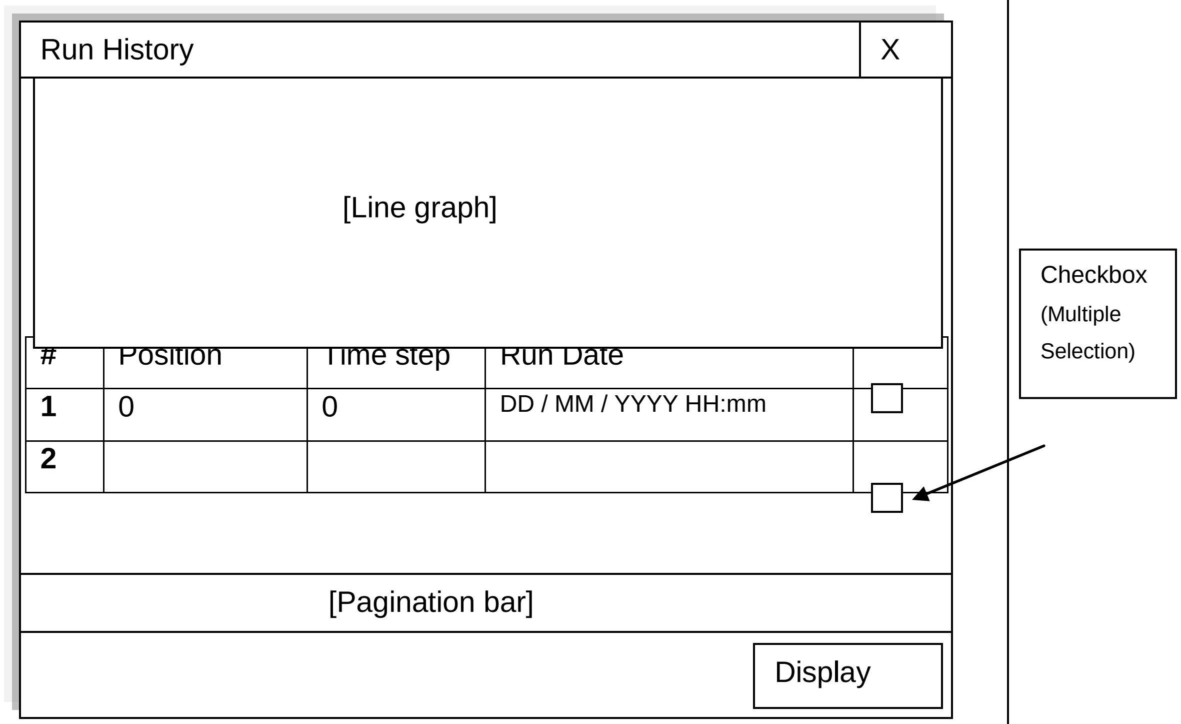

The rule definitions form will be a child of the main grid.

Ant Rules

	State	Direction
	[Colour Picker]	Left, or right – Drop down list.

History Form

The history form will be a child of the main grid.



#	Position	Time step	Run Date	
1	0	0	DD / MM / YYYY HH:mm	☐
2				☐

Requirements Analysis

Written Use Cases

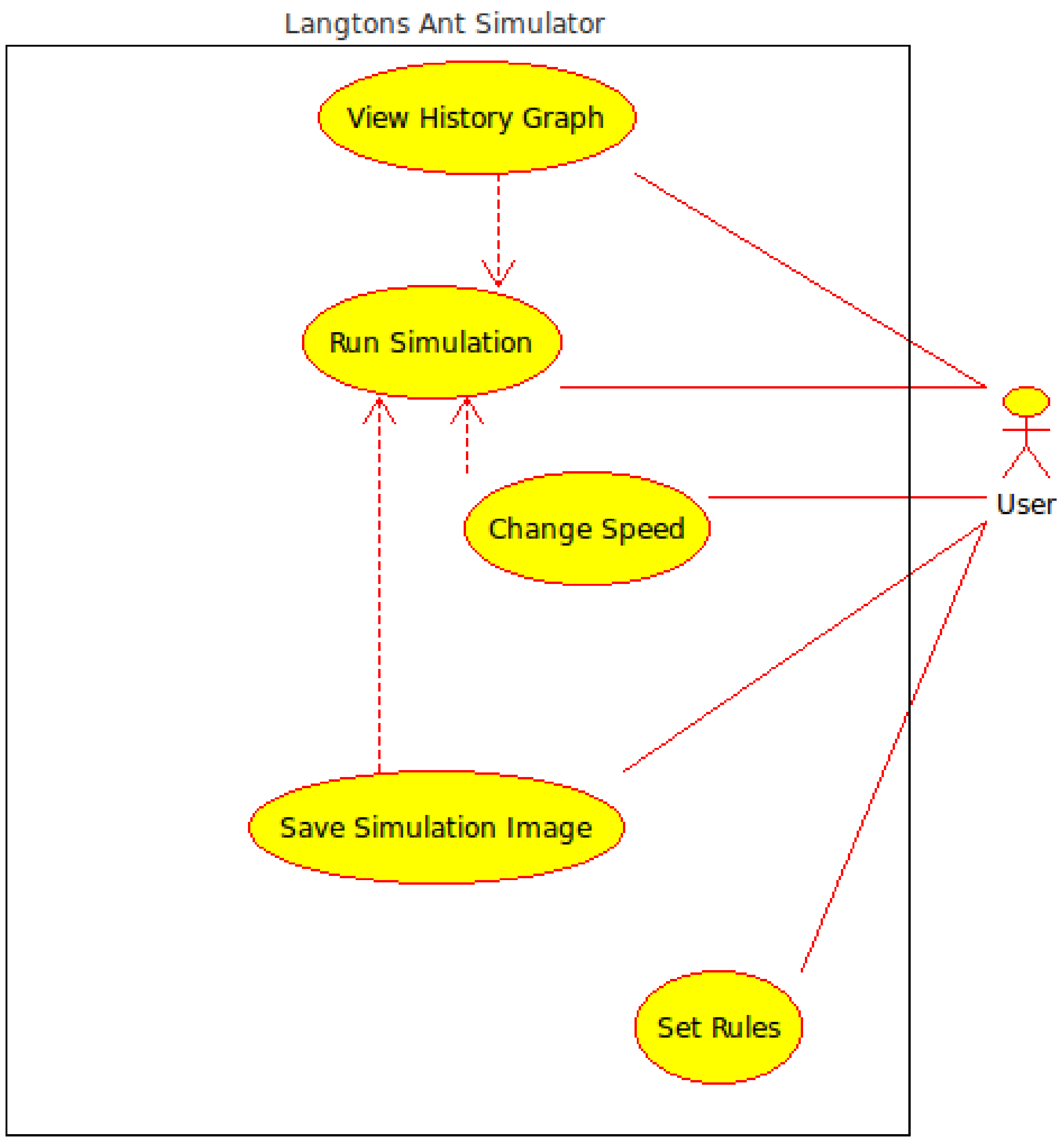
Use Case:	Run Simulation
Summary:	The user will run the Langton's Ant simulation.
Related Cases:	Set Rules
Actors:	The User
Pre-conditions:	The user must have launched the program. **Optional:** The user may have set rules for the ant to follow.
Post-conditions:	The simulation will have run.
Steps:	**1.** The user navigates to the start simulation button. **2.** The user pressed the start simulation button.

Use Case:	View History Graph
Summary:	The user will view the graph of the user's movement.
Related Cases:	N/A
Actors:	The User
Pre-conditions:	The current simulation must have finished. There must be at least one simulation in the database.
Post-conditions:	A graph depicting the ant's movement will have been generated.
Steps:	**1.** The user chooses a past run to generate the graph from and it is generated.

Use Case:	Save Simulation Image
Summary:	The user saves the final image at the end of a simulation.
Related Cases:	Set Rules
Actors:	The User
Pre-conditions:	The simulation must have finished or been stopped by the user.
Post-conditions:	The image will have been saved to the hard drive.
Steps:	• (**optional**) The user stops the simulations. • The simulation ends. • The user saves the generated image to their hard drive.

Use Case:	Change Speed
Summary:	The user will change the speed of the simulation.
Related Cases:	N/A
Actors:	The User
Pre-conditions:	There must be a simulation currently running.
Post-conditions:	The speed of the simulation will have changed.
Steps:	• The user navigates to the speed slider. • The use alters the speed of the simulation by moving the slider.

Use Case Diagram



Functional Requirements

The functional requirements defined below show the internal workings of the software that is being developed. That is, specifically states the technical details, data manipulation, graphing, and other specific functionality of the Langton's ant software.

Name:	**FR-1.0.** Langton's ant solution.
Summary:	Visually shows the ant simulation as it progresses through the pre-defined rules.
Rationale:	The user will choose to run the ant simulation, providing it with a set of rules that it will follow. This will produce a set output.
Requirements:	The ant simulation will initially be run on a grid of 100x100. The ant will then progress through a set of rules outlined prior to the simulation and each step it will perform the said task. That is the ant will move forward one and if the square is black, it will then turn right, invert the colour of the square and move forward to the next one. The progress will of the ant will continue indefinitely until it stopped.
References:	N/A

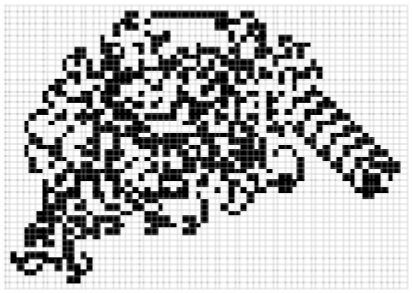


Figure 1: The result of an ant simulation.

Name:	**FR-1.1.** Grid.
Summary:	This determines the properties of the grid.
Rationale:	The user may wish to change the size and colour of the initial starting grid that the ant simulation will be running on.
Requirements:	In the options section of the software, there will be an ability change the colour and size of the grid. The default values of the grid will be a back colour of brown and size dimensions of 100x100. During the course of the ant simulation the software will increase the size of the grid if the ant goes towards the edge of the grid.
References:	**FR-1:** Langton's ant solution.

Name:	**FR-1.2.** Rule layout.
Summary:	The rules that the ant follows during the course of the simulation will be displayed on screen.
Rationale:	The user may wish to know what the ant is doing for which square. It will be there all the time for look-up.
Requirements:	The rules applied to the current ant simulation will be displayed during the course of the run. It will be list based on either the left or right hand side of the GUI. Each line in the list will indicated the colour of the square and the ant movement that will result upon hitting this square.
References:	**FR-1:** Langton's ant solution.

Name:	**FR-1.3.** Start.
Summary:	This will start the ant simulation.
Rationale:	The user will be able to initiate the ant simulation by clicking the start button.
Requirements:	The software will need to be able to initiate the ant simulation upon the start button being clicked.
References:	**FR-1:** Langton's ant solution.
Name:	**FR-1.4.** Stop.

Summary:	This will stop the ant simulation.
Rationale:	The user will be able to stop the ant simulation whenever required by clicking the stop button.
Requirements:	The software will need to be able to stop the ant simulation upon the stop button being clicked.
References:	**FR-1:** Langton's ant solution.

Name:	**FR-1.5.** Save simulation image.
Summary:	The image of the final look of the simulation is saved to the hard disk (**Figure 1**).
Rationale:	The user may wish to record the ant simulations as an image for later reference.
Requirements:	When the simulation finishes, or stops, the user is given the option to save the final image to the hard drive.
References:	**FR-1:** Langton's ant solution.

Name:	**FR-1.6.** Change speed.
Summary:	Allows for the ability to change the speed of the simulation during run-time.
Rationale:	The user may wish to increase the speed of the simulation.
Requirements:	A sliding bar at the bottom of the GUI will allow the user to dynamically change the speed of the ant simulation during run-time. It will do so by increasing the amount of steps the ant takes at a single instance.
References:	**FR-1:** Langton's ant solution.

Name:	**FR-1.7.** Saving solution to database.
Summary:	The results of the ant simulations will be stored in a database, so the data can be used in future.
Rationale:	The data stored will enable the user to generate line graphs for easy data interpretation.
Requirements:	Upon the completion of an ant simulation, the resulting data from that run will be stored in a database. The data stored will be the step number and the ant position at that point in time. This data is stored for each time step that the ant goes through in a single simulation. Could be as many as 80,000 steps.
References:	**FR-1:** Langton's ant solution, **FR-3.0.** Database development.

Name:	**FR-2.0.** Rule definition.
Summary:	Rules can be applied to the software to create different results for the ant simulations.
Rationale:	The user may wish to give the ant simulation a different set of rule strings. Causing a diverse range of output results.
Requirements:	In the options menu of the software, there will be an ability to specify new/different rule strings for the ant simulation. These rule strings will be stored in the database in conjunction with the other data stored for that particular simulation. The rule data the user can specify will be the square colour and the ant movement once upon this square.
References:	**N/A**

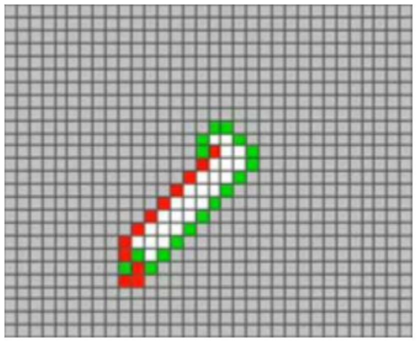


Figure 2: Possible look of an ant simulation given three rules.

Name:	**FR-3.0.** Database development.
Summary:	The database used to store all the relevant data for this software application.
Rationale:	Whenever the user uses a function that requires stored data, it will be retrieved from here.
Requirements:	The database will be developed using MySQL and will store all relevant information for the ant simulations. It will be tied into the software to be utilised with the various required functions for data storage and retrieval.
References:	**N/A**

Name:	**FR-4.0.** Graphing.
Summary:	Generates a line graph based on user selected data.
Rationale:	User selects previously done simulations to generate a line graph.
Requirements:	Line graphs will be generated to show the data based on one or many ant simulations. The data of these will be passed to the graphing utility through the simulations selected by the user. The Y-axis will be for the time step and the X-axis will be for the ant position.
References:	**N/A**

Name:	**FR-4.1.** Graphing based on selected runs.
Summary:	User selects ant simulation results from the simulation history (**FR-5.0**) to generate a line graph on.
Rationale:	User selects one or more simulations to compile a line graph on.
Requirements:	The software needs to be able to generate a line graph based on multiple ant simulation results. These will be gathered from the ones the user has selected and passed through to the graphing utility.
References:	**FR-4.0.** Graphing.

Name:	**FR-5.0.** Display history of runs.
Summary:	Displays a history of all the Langton's ant simulations that have been run and stored in the database.
Rationale:	The user requests all the data stored on all the different simulations that have ever been initiated.
Requirements:	The software will be tied into the database table that stores the history of all the ant simulations. When the user requests the simulation history, each record will be returned and displayed in a data grid, or grid, with a paging index to limit results. Results will be filtered by date with the newest simulations first.
References:	**FR-3.0.** Database development.

Name:	**FR-5.1.** Select multiple runs for graph output.
Summary:	Multiple graphs can be selected to be utilised as data for the display and calculation of a line graph.
Rationale:	The user will select various output results for the generation of a line graph.
Requirements:	The history displayed (**FR-5.0**) will have a check box associated with each record. This will enable multiple output results to be selected. Once selected the data from these runs will be compiled and utilised to generate a line graph. The user can select 1+ runs to generate a graph.
References:	**FR-5.0.** Display history of runs, **FR-4.1.** Graphing based on selected runs.

Non-Functional Requirements

Name:	**NFR-1.0.** Python programming environment
Summary:	Python must be on the target machines.
Rationale:	Python needs to be installed on the machine the program is run on, without it the program will not run.
Requirements:	It was stated by our sponsor that the program was to be written in Python.
References:	**N/A**

Name:	**NFR-2.0.** Professional development
Summary:	Developers must learn Python
Rationale:	The Developers need to be able to program in python.
Requirements:	Because the program is being written in Python the developers must learn python to finish the program.
References:	**N/A**

Name:	**NFR-1.0.** Programming constraint
Summary:	Must be developed in Python
Rationale:	The program must be written in Python.
Requirements:	It was stated by our sponsor that the program was to be written in Python.
References:	**N/A**

Software Development Plan

Scope statement

Project Justification: The project is initially a sponsored software research undertaking; which will investigate, and coordinate the development of a feasible solution for Langton's ant. The key aspect in this project is to build a functioning system that will demonstrate the operation of a self-replicating cellular automaton.

.

Product Characteristics and Requirements:

1. The application should contain dynamically scalable grid; which should resize independently based on the ant's position, therefore, providing the ability for the ant to traverse for an infinite amount of steps (based on the given rules).
2. The rules will have an ability to be defined, and therefore allowing for the ant to operate on a number of cellular states (not just the default two).
3. The ant will have the ability to be 'controlled' by the user; in the respect of encompassing the function to initiate/pause and stop the time steps, also comprising the ability to increase the time speed. The time speed will also be displayed; providing the user with knowledge of the current velocity.
4. The ant's escapade will be logged and saved into a database, which will also provide the user with an ability to view the past runs. The history will be presented to the user in a separate child window; enclosing a data-grid, and a mutable line graph. The line graph will be redrawn based on the 'history' selection from the grid, and show ant's journey based on the grid cell number (y-axis) and time-step (x-axis).
5. Once the user has terminated the current run cycle; the user will be prompted with the option to save a screen capture of the generated pattern. The image however will only be a savable feature, and will not later redisplay the saved image in the history section.

References

Wolfram, S. (2002). History of cellular automata. Retrieved March 12th, 2010, from http://www.wolframscience.com/reference/notes/876b

Langton’s Ant. (2010). Retrieved March 12th, 2010, from http://en.wikipedia.org/wiki/Langton%27s_ant

A new kind of science. Retrieved March 13th, 2010, from http://en.wikipedia.org/wiki/A_New_Kind_of_Science

Langton’s Ant. (2010). Retrieved March 14th, 2010, from http://mathworld.wolfram.com/LangtonsAnt.html

Moreira, A., Gajardo, A., & Goles, E. (2001). Dynamical Behavior and Complexity of Langton’s Ant. Complexity, 6(4), 46-51.

Langton, C. G. (1986). Studying Artificial Life With Cellular Automata. Physica D: Nonlinear Phenomena 22(1-3), 120-149.

Aldoaldoz. (2009, January 18). Langton’s Ant [Video file]. Video posted to http://www.youtube.com/watch?v=1X-gtr4pEBU

Paul Grobstein (2005), The World of Langton's Ant, Retrieved March 14 2010, http://serendip.brynmawr.edu/complexity/models/langtonsant/index5.html.