Teaching Old Turtles New Tricks: Artificial Life Simulation Using Lingo

Andrew M. Phelps
Information Technology Dept.
College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY, 14623
http://andysgi.rit.edu/

Daniel R. Kunkle
Information Technology Dept.
College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY, 14623
http://www.rit.edu/~drk46633/

Abstract

The new and expanding interdisciplinary field of artificial life (Alife) is not yet well defined and will mean various things to different people. Alife.org [2000] states broadly "The term Artificial Life is used to describe research into human-made systems that possess some of the essential properties of life." This article details the simulation of one such life-like system implemented using Macromedia Director. Though many of the ideas and techniques relating to alife that are explored in this project have been studied for many years, this project breaks new ground as it uses Director to allow the user to easily visualize the life-like processes occuring within the simulation.

Speciffically, this project aims to explore the visiualisation of two components of complex, life-like systems: decision making and evolution. This article contains four main sections, the first describing the simulation interface and it's components, the second detailing the underlying representation of individual life forms and their environment, the third exploring the mechanism for decison making along with other life like features and the last concerned with the use of a genetic algorithim for the evolution of artificial life. Additionally, possible future explorations for the project are proposed.

1 THE SIMULATION ENVIRONMENT

1.1 TUTORIAL FILE SETUP

This project was developed through a seminar in artifical life involving faculty and students of the Department of Information Technology at the Rochester Institute of Technology. It's main goal is an intorduction to the simulation of life-like functions in an artificial environment.

To get started, open the demo (alife.dir) that accompanies this article. Starting the movie will create a number of artificial life forms and start the simulation. This group of life forms is the first generation. They will begin performing actions based on their genetic code. These actions may involve changing their movement speed and direction based on a number of percieved environmental conditions and consuming food to survive.

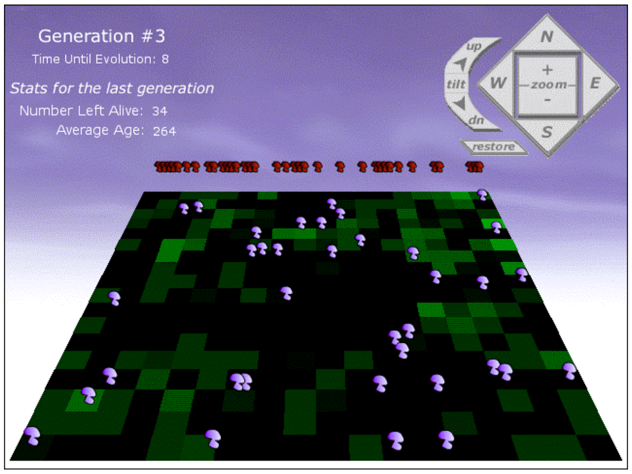

Figure 1: View of simulation, running with default parameters and variable setup.

1.2 VISUALIZATION OF LIFE

The moving, blue forms represent the living individuals. They are sometimes refered to as "Turtles" because of their Logo based heritage. The plane of green and black squares represents the land that the individuals move on. These squares are termed "patches". This term come from StarLOGO [MIT, 2001], the simulation tool on which this project's environment is based. The more green a patch has the more food it contains. A turtle will eat when it is on a patch that contains food. Eating is the only way a turtle can increase it's energy level. A turtle is constantly using energy to maintain the functions of life and will spend energy faster when it moves more quickly.

So, a turtle must spend energy to move to find food to get more energy, just like real life. If a turtle is unable to find enough food it's energy level will reach zero and it will die. When a turtle dies it becomes red and is placed in the "graveyard" at the back of the environment.

After a certain amount of time the simulation halts and reproduction takes place. The population replenishes itself with a new generation of turtles and the simulation restarts. The generation number and the time left until the next generation will be generated are displayed in the upper left along with statiscs detailing how well the last generation fared, including the number of individuals that survived the last generation (out of 72) and the average age of all of the individuals at the end of the last generation (a maximum of 350 if all individuals survive to the end).

This simulation is set to start with no auto-matic rebounding behaviour to keep the turtles on the board. In fact, you may notice a substantial number of them flying off in all directions in the first few stages of exolution. Over time however (10-20 generations) you should notice that many of the turtles **develop their own rebounding behaviour** (as well as a number of other behaviours which this paper will describe). Such was our intent – to allow an object to develop behaviour without explicitly programming than model into the object. Turtles rebound because by doing so they can continue to get food, and live longer, both of which are goals of the overall ecology.

The controls in the upper right change the users view of the world. Allowing them to move and zoom their view relative to the scene. Experiment by watching turtles close up to note their individual difference, and by zooming out to note the trends in the entire population.

2 SIMULATION ENGINE

2.1 PATCHES AND TURTLES

The entire basis of this simulation revolves around two very important concepts, the "turtle" and the "patch". The "turtle" traces its heritage back to the M.I.T. "Turtle Graphics" work, which was first presented by Abelson and diSessa [Ableson & diSessa, 1980]. Also of interest are the similarities of this work to the "vehichles" presented by Braitenberg [1984, 1996]. A "turtle" has the following characteristics: it can perform a limited number of self-contained action, and in the original implementation, these were geometrical. Examples of simple actions were things like `'move()'` which would move a turtle along its forward vector the number of units specified by its speed. Additionally it could turn right or left, set its speed to a different value, and (by combining these actions) rebound itself from collisions. The turtle can be thought of in programming terms as the collection of a local coordinate system within an object, with the methods of that object providing manipulation of that coordinate axis. Additionally, turtles usually had the ability to represent themselves, as some element that is drawn to the screen, in our implementation we have given the turtle a sprite property, and it uses that sprite to represent each turtle, or instance of the object (see figure 2).

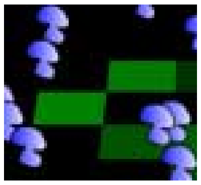

Figure 2: A few instances of the turtle object visualized as blue 'mushrooms' on the ground.

The second important notion then, once a world is filled with turtles, is the environment that the turtle is living in, and in this simulation we borrow heavily from the notion of a "patch" used in the StarLOGO environment [MIT, 2000]. A patch is a plot of ground that stores within itself certain attributes, and again we implement this structure as a Lingo object. Thus a patch is really a container for variables that describe the environment at that particular geometric space. In more complex environments the patches can be though of as three-dimensional bins rather than two dimensional planes, a turtle in our world could feed off a patch regardless of how high in the air it was floating, we only check the X and Z coordinates (Y axis is vertical up / down).

The interesting thing to note about this system is that, like its predecessors, the turtle can only know about its current patch and the neighboring patches by comparison (for example, it may be able to deduce 'the patch to my immediate North has more food than my current patch). They do not receive a 'global view' of the world any more than we as individuals do. Indeed just has hard-coding behavior is relatively simple, so to is it trivial to simply give the turtle complete knowledge of everything and anything to do with its environment, and then expect the evolution of complex behavior. If a turtle knows everything, it will simply hop from its current square to whichever square has the most food that it can travel to, and it will know if it can make it because of its god-like knowledge. Such a system was uninteresting to the authors, as it did not allow for multiple strategies to emerge and be incorporated into the general solution.

2.2 3D VIEWING OF A 2D WORLD

This world of patches and turtles then, could be presented from a top-down point of view, similar in some sense to the StarLOGO environment [MIT 2001]. Instead, the decision was made to leverage an existing Director project, the LingoLand 3D Engine [Phelps 2001]. This engine, while not wholly optimized for real-time performance, ran well enough to offer a perspective view of the simulation environment. The environment is made up of patches, which in turn are represented by the quads drawn by the engine. These tiles are expanded from the previous work of the terrain simulation to include information about the amount of food and other information necessary for the turtle / tile interaction within the tile data structure.

The 3D engine at the heart of this environment operates on a few simple principles, and by using a number of methods that are listed here (see Table 1), and which are described more fully in the paper describing LingoLand [Phelps 2001]. First, we have linked the control of the engine's camera to the buttons on the upper right of the screen, allowing the user to manipulate the scene. This manipulates the gCamera's properties for its X and Y and Z position vector, though its associated move and roll methods. This camera is used in the projection of the tiles upon the stage, causing the scene to appear in relative perspective. Perhaps the most useful feature of the camera is its ability to 'zoom in' on the board, offering a closer view of turtle behavior during the simulation.

Table 1: Engine Control Variables (partially reprinted from LingoLand: Simple 3D Terrain Simulation in Lingo for ease of reference).

VARIABLE	DESCRIPTION
max_row max_column	The size of the original land matrix. This is the number of 'points' in X and Z that you see the original land created with when the movie starts.
square_size	Size in pixels of the original tiles, the space between the points defined above.
gCamera	The viewpoint for the scene, see section 1.1 and 1.2 for details.
gZoom	Original zoom level used by the projection matrix relative to the camera.

Removed from the original engine is support for dynamic lighting, as this did nothing to augment the simulation, and in fact made it more difficult to read the color-coding scheme that describes the desirability of the patch. Also removed from this demo is the code that randomly assigned an elevation to the ground, producing the original output of the terrain simulation (see Figure 2).

The removal of these features was simply for the ease of student use and the requirement to complete the simulation within an academic quarter. Nathaniel Swart (Nathaniel_M_Swart@firstclass.it.rit.edu) led a student team that did use elevation within the simulation, using elevation as a criterion for food growth and as a criterion that helped determine the ease with which the turtles could move from patch to patch.

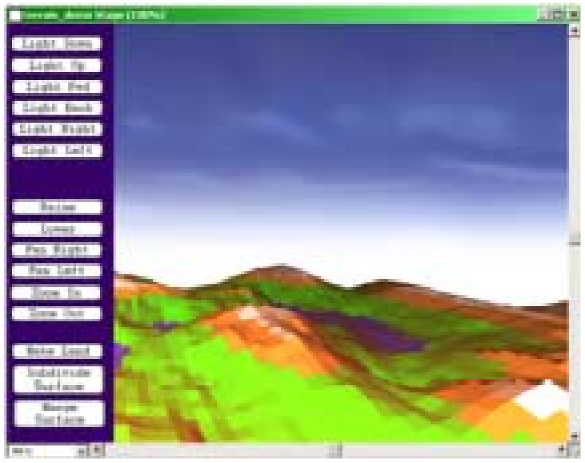

Figure 2: Original Terrain Simulation Engine. Lighting and Elevation were removed for the A-Life Simulation Demo.

2.3 MOVING AND TARGETING

The movement of turtles is somewhat complex, and is described in its most basic sense by Kurtz in his original 2D implementation [Kurtz, forthcoming]. The basics of the transformation of that engine to the engine contained in this project are presented by Phelps in *LingoLand* [Phelps, 2001]. A turtle contains a series of methods that allow it to target a number of different types of locations within the world, and a simple move method that moves a turtle along its pFwd vector. The movement and targeting methods are summarized in Table 2, and are commented in the Lingo Code.

Table 2: Turtle Movement and Targeting Methods. See the VBLF script for implementation comments.

METHOD	DESCRIPTION
new	Create a VBLF.
print	Print the location vector of a VBLF
RollLeft	Roll counter-clockwise along the pForward vector, X-Axis upon creation.
RollRight	Roll clockwise along the pForward vector, X-Axis upon creation.
PitchUp	Roll counter-clockwise along the pLeft vector, Z-Axis upon creation.
PitchDown	Roll clockwise along the pLeft vector, Z-Axis upon creation.
YawLeft	Roll counter-clockwise along the pUp vector, Y-Axis upon creation.
YawRight	Roll clockwise along the pUp vector, Y-Axis upon creation.

Table 2: (continued)

VARIABLE	DESCRIPTION
Move	Move a VBLF along its forward vector a unit equal to its speed property.
MoveTo	Reposition a VBLF to an absolute coordinate value.
VBLF_Project	Reset the h and v properties of a VBLF so that it draws on the 2D stage correctly.
HeadForVect	Orientes the sprite to head for the specified position vector.
HeadForPt	Head for the specified 3-D point (x,y,z)
HeadForVBLF	Head towards a the position of a second VBLF
X, Y, Z	Returns the x, y, or z coordinate of the turtle.
TurnTo	Turn to an absolute heading in degrees. Currently only implemented for the XZ plane.
Distance	Returns a position vector between the vblf and a target position.
DSquare	Same as distance but does not use sqrt in calculation. Distances are relative not scaled to the coordinate space.

3 MAKING IT LIFE-LIKE

3.1 OVERVIEW OF ARTIFICIAL LIFE

Artificial life, most broadly, is life created by human effort rather than natural occurrence. This covers a massive range of study and must be narrowed in this case to the aspects of artificial life that this project addresses. Resources for artificial life in general can be found in the annotated bibliography.

In our case, artificial life is the simulation of life using computational methods. This can have two obvious uses: simulating existing biological systems to gain a better understanding of them and simulating new systems that have no natural analog. The first use is becoming more and more widespread and there are a number of tools available that allow this kind of explanatory simulation, including StarLogo developed at the Media Laboratory, MIT. StarLogo is also used as an educational tool to teach students about the modeling of decentralized systems [MIT, 2001].

Our simulation has no natural analog. It is not meant to shed light on some existing system, but rather to create a novel environment that has some of the features of a natural system. This approach is becoming increasingly popular in computer gaming where the possibility of a game that conforms to its players and evolves over time is particularly appealing.

This project simulates two main features of living things, a decision making process to choose possible actions based on current conditions, and evolution to adapt to an environment. The implementation of these features is covered in the following sections.

3.2 DECISION MAKING PROCEDURE

Decision making is a trademark of life, and a complex one at that. What a given life form does at any moment is based on a great number of things, and these things can be separated into two categories: internal and external. The state of the individual and the state of the environment as they perceive it determines what they will do next. This procedure is modeled in the simulation in a simplified form.

Becuase the traditional turtle from logo can do little else than take movement instructions, these upgraded individuals are refered to as Decicision Making Turtles (DMTs), and are the creatures you will find roaming the simulation.

At any given moment the DMT will find itself in one, and only one condition. This condition is based on properties of both the DMT itself and it's immediate surroundings. In the basic simulation there are only two properties, one internal and one external: (1) The DMT’s energy level, and (2) whether or not the patch the DMT is currently on has food or not.

These properties are defined as binary digits, each having only two possible values. For the first, the DMT either has high or low energy (high and low being defined at the start of the simulation). The second, food is present or not present. This leads to four possible conditions, (1) low energy/no food, (2) low energy/food present, (3) high energy/no food and (4) high energy/food present. Replaced by binary digits these conditions are 00, 01, 10 and 11 (or 0, 1, 2 and 3 in decimal). So, a DMT always finds itself in one of those four conditions. This covers one half of the decision making process, a DMT must now decide what actions to perform for the given condition.

Along with the list of conditions, a list of actions that a DMT has available to it is specified. In this simulation that list includes thefollowing:

(1) nothing - DMT performs no action

(2) speed up - DMT speeds up one speed unit

(3) slow down - DMT slows down one speed unit

(4) stop - DMT sets speed to zero

(5) head for food - DMT heads for the neighboring patch with the most food. Each patch has 8 neighboring patches. If the patch that the DMT is on has more food than any of those 8 the DMT does nothing.

(6) turn around - DMT turns 180 degrees

(7) turn random amount - DMT turns a random amount, from 1to 360 degrees.

(8) head for neighboring turtles - DMT heads for a the neighboring patch with the most DMTs on it. If the patch that the DMT is currently on has more turtles than any of the surrounding 8 patches the DMT will do nothing.

(9) head away from neighboring turtles - DMT heads directly away from the neighboring patch with the most DMTs on it. If the patch that the DMT is currently on has more turtles than any of the surrounding 8 patches the DMT will do nothing.

Notice that "move" is not an action. This is because each DMT will move each frame. If the DMT’s speed is zero then moving will not change that DMT’s location.

The only thing left is to connect the conditions to the actions. Each DMT is given a "genome", which is a list that specifies the actions to be performed in each condition. In this simulation, a DMT can perform up to 2 actions in any given condition. At the beginning, each DMT’s genome is randomly generated, having either 1 or 2 random actions per condition.

An example random genome: [3,4] [7,3] [5] [4]

The first set of []'s corresponds to the first condition listed above, low energy/no food. So, when a DMT has low energy and is not on a patch with food it will first slow down one speed unit, then stop. It is possible for actions to cancel each other out when performed in the same condition, such as speed up and slow down.

These conditions and actions could easily be modified and augmented to provide the DMTs with goals and considerations other than simply finding food to stay alive. In one version of this simulation there were a number of red patches scattered around that were designated as “goals“ that DMTs would consider in their decision making process.

A DMT’s genome will not change during their lifetime, so DMTs with a poor genome will die earlier than those with a more advantageous genome. The population of DMTs, however, can improve through evolution.

3.3 EVOLUTION

The accepted method of evolution in nature is through natural selection, as formulated most famously by Charles Darwin. The theory of natural selection is based on a few assumptions and ovservations: (1) The environment provides limited resources that individuals must compete for (2) Individuals have differences (3) Individuals can pass on their differences to offspring. This leads to the inference that some individuals, because of their differences, will be more able to acquire those limited resources and therefor survive more easily and reproduce more easily. These offspring in turn will acquire some traits from their parents that will allow them to survive more easily, and so on. Over time, evolution.

This is just the senario that has been created in this simulation. The environment has limited food resources. Individuals differ through their mapping of conditions to actions. And, through their genome, better performing individuals can pass their traits onto their offspring.

3.4 COMPLEX BEHAVIOUR FROM SIMPLE BRAINS

The use of evolution in this simulation resulted in emergen behaviors. Emergent behaviors are complex ones that result from the interation of some simpler actions.

In this case, we programmed the DMTs with a small set of simple actions, but didn’t explcitly define how they are to be used. The optimum use of these actions was determined by the DMTs evolution in their environment.

One fairly important emergent behavior, as far as survival is concerned, is grazing. In the first generation, the DMTs often run wildly, wasting energy, or stand still, failing to find food sources. In later generations many of them seem to graze the environment, moving to patches that have large amounts of food and moving on once those patches are delpleted. This behavior was in no way defined in the simulation but yet appears consistantly with several runs.

Another emergent behavior observed was rebouding, or turning around upon reaching the edge of the patches. This behavior was explicitly programmed into the system, as it is in any simulation where characters can move about in a finite area. In this case though the rebouding doesn’t always work. DMTs, if they are stuborn enough, can continue to move off the edge of the patches. The rebounding programmed into the simulation could most likely have been improved to insure no DMT could go off the board, but evolution provided for it after a number of generations. This is because DMTs that leave the board have no way of obtaining food and so usually die-off sooner. Through evolution these individuals reproduce less often and DMTs stay within the area of the patches more regularly. This was tested by removing the built in rebouding and observing the results. To experiment with this yourself, make sure that the global variable `gReboundingOn` in `StartMovie` is set to false, and let the program cycle through 10-15 generations. You should see a marked decrease in the number of turtles that fly off the board over time.

4 GENETIC ALGORITHM

4.1 GENOME REPRESENTATIONS

The first step in implementing a genetic algorithm (GA) is representing the thing you want to improve with a genome of some sort. In this case, since it is the decision-making effectiveness of the individuals that we want to improve, the genome used in the genetic algorithm is the same as that used for decision-making. That is, a list of actions for each condition.

4.2 GENETIC ALGORITHM PROCEDURES

The procedures described here are based on Goldberg's explanation in his first chapter an "Introduction to Genetic Algorithms“ [Goldberg 1989]. Much work has been done on GAs in the last few decades along with a number of good introductions for those new to the subject.

A genetic algorithm selects those DMTs that have performed better (i.e. ones that have lived longer) and mates them, producing a new generation of DMTs. This new population is produced by halting the simulation momentarily and performing the following steps:

(1) Calculate each DMT's fitness - DMTs that have lived longer by feeding more will have a higher fitness. In more complex simulations fitness would be based on a number of criteria.

(2) Create a mating pool - This mating pool is half the size of the original population. Each member of the mating pool is selected randomly from the current population, with DMTs that have a higher fitness being more likely to be selected.

(3) Reproduction through crossover - Each member of the mating pool is mated with one other, so that each DMT that made it to the mating pool reproduces with one and only one other DMT from the mating pool. Their decision making genome are split at a random point, crossed over, and produce two offspring (see Figure 3).

Parent1 = [1,2] [6] [9,4] [5]

Parent2 = [4,4] [3] [7,2] [9,8]

random crossover point = 3 (between third and fourth conditions)

offspring1 = [1,2] [6] [9,4] [9,8]

offspring2 = [4,4] [3] [7,2] [5]

Both parents and both offspring are then placed into the next population.

Figure 3: a simple example of cross over.

(4) Mutation - each child has a small random chance that it's genome will be slightly altered through mutation. This helps to encourage diversity in the population, hopefully leading to better genome that might not otherwise have been discovered. It is the only way in this simulation to achieve new combinations of actions within a condition, as crossover only acts on combinations of sets of actions.

(5) Restart the simulation and repeat steps 1 - 4 at set intervals(about one generation every 1 minute, depending on the speed ofyour machine).

After a number of generations, the DMTs will begin to find food better and live longer.

To track the evolution of the DMTs, the observer can watch the statistics in the upper left of the screen. The basic simulation will give two statistics: (1) the number of turtles left alive at the end of the generation and (2) the average age of the population. Both of these will increase as new generations are created.

5 CONCLUSIONS

This project met a number of its original goals. The simulation contains life-like features, namely non-deterministic decision-making and evolution. Equally important was the fact that these features were easily observed and understood by those with no acquaintance with the underlying subject matter.

From a developer's perspective, one of the most satisfying things was having the simulation develop unexpected, emergent behaviors and having these behaviors be positive ones. This is opposed to most unexpected behaviors in other applications, which are almost always bugs needing to be fixed. Often, the more life-like something is the more it can adapt and grow without explicit instruction and intervention.

The educational benefits of teaching the Artificial Life class using Director were exceptional: Director offered a platform that was completely open, and had the necessary capabilities for this rudimentary model, while still offering students a familiar background against which to work. Most often courses in artificial life are first taught in Lisp, or sometimes in C / C++. This is usually done to accommodate the eventual need for recursive strategies, or, in some cases, code modification on the fly. However, with the advent of more modern languages, these languages are now typically not taught as part of the normal student sequence, with R.I.T. and many other institutions moving to Java. Director, however, manages to provide almost all the necessary functionality, without encapsulating the routines themselves such that students don't feel 'programmatically uninvolved', which was some students' reaction to the StarLOGO environment.

Second, the course was offered as a seminar, which was a structure that fit part and parcel with the idea of Artificial Life. Each member of the class brought something unique to their way of thinking about the research, and without a set syllabus, it was possible to capitalize on many of the directions that students wanted to follow. Much of the direction of the course was set by my co-author Dan Kunkle, who at a very early stage in the course began to think of the Turtles as having actions described by genomes. There are certainly other ways of implementing a genetic algorithm, but he was the most precise in his presentation of a possible direction, and the rest of the class followed this direction behind him.

The final student experience for that course was positive, although the course received mixed reviews [R.I.T. 1999].

Based on personal observation it would appear that the students who more fully explored the research of Artificial Life as a field felt extremely satisfied with the course and the final implementations, while those that were looking for a more traditionally structured course felt less satisfied, although the course evaluation metrics available at R.I.T. are incapable of either supporting or disproving this hypothesis. We note this for the sake of other educators who may wish to use the material to found an introductory course to Artificial Life for a similar class of students: the incorporation of genetic algorithms and the evolutionary approach will take time for students to grasp, some of whom will become exceedingly frustrated when they feel they could just hard-wire the desired behavior with greater ease. This project will likely be presented more formally from a pedagogical point of view in the future, but for this paper, which is technically oriented, it should be noted the response that these projects received.

If you are considering using this material for classroom dissemination, please contact the author(s) for additional files, materials, notes etc.

6 FUTURE WORK

Needless to say this engine could be developed further, and the underlying structure has almost limitless possibilities in allowing sprites to develop unforeseen or optimized behavior, as ours developed a simple rebounding structure. This is optimal for larger solutions in which the optimal behavior for a sprite is unknown, or for situations in which it is desirable for the sprite to demonstrate some appearance of intelligence, computer games offering one of the more exciting possibilities. The authors hope to expand and extend the work presented here, possibly producing milestones and additional algorithmic implementations along the way.

Acknowledgments

The authors would like to thank Steve Kurtz for his participation in developing and teaching the course, as well as Bruce Damer and all the rest who started the Digital Biota Working Group, and Karl Sims, who inspired us all. Additionally, this project could not have been successful without the student body of the Information Technology Department at the Rochester Institute of Technology. Thanks to everyone who took the chance and signed up for the course.

References

Abelson, Harold and Andrea diSessa. (1980) *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. Cambridge, Massacheusetes: The MIT Press.

A. Liekens (2000): "Artificial Life?", in: A. Liekens (editor): *Artificial Life Online 2.0*, URL: http://www.alife.org/index.php?page=alife&context=alife.

Braitenberg, Valentino. (1986, 1995). *Vehichles: Experiments in Synthetic Psychology*. Cambridge, Massacheusetes: The MIT Press.

MIT Epistomology Group, Media Lab, Massechusets Institute of Technology. *Introduction to StarLogo*. (2001) Online: http://el.www.media.mit.edu/groups/el/Projects/starlogo/index.html.

Steven Kurtz and Andrew M Phelps. *Vector Based Life Forms, a 3D Engine Bawsed on Turtles*. Forthcoming. Featured Article in Using Director – Director Online. Online: http://www.director-online.com/

Goldberg, David E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massechusetts: Addison-Wesley.

Steven Kurtz. Turtle World. (Forthcoming) Featured Article in Using Director – Director Online. Online: http://www.director-online.com/

Phelps, Andrew M. *LingoLand: Simple 3D Terrain Simulation in LIngo*. (April 2001). Featured Article in Using Director – Director Online. Online: http://www.director-online.com/

Annotated Bibliography

[GA]Genetic Algorithm; [L] Lingo based 3D Code; [M] Mathematics; [AL] Artificial Life; [G] Graphical Representation.

1. Edgerton, P.A & W.S. Hall. (1999) *Computer Graphics: Mathematical First Steps* Essex, England: Prentice Hall. [M]
2. Lithium. (1999-2001) Three Dimensional Rotations. Online. http://www.gamedev.net/ [M]
3. Rodgers, David F. (1985) *Procedural Elements for Computer Graphics*. New York, New York: McGraw Hill. [M][G]
4. Tamahori, Che. (1999) How to Cook 3D in Director. Online. http://www.sfx.co.nz/ tamahori /thought/ shock_3d_howto.html. [L][M][G]
5. Watt, Alan and Fabio Policarpo. (2001) *3D Games: Real Time Rendering and Software Technology*. New York, New York: Addison-Wesley ACM Press. [M][GA]
6. Zavatone, Alex. *Inside Zavs Brain: 3D Director*. Online. www.director-online.com/accessArticle. cfm ?id=286. [L]
7. Swan, Barry. (2000) T3D Engine. Online. http://www.theburrow.co.uk/t3dtesters/. [L][G]
8. Langton, Christopher G. (1995, 1997) *Artificial Life: An Overview*. Cambridge, Massacheusetes: The MIT Press. [AL][GA][G]
9. Langton, Christopher G., Richard K. Belew, Hiroaki Kitano, and Charles E. Taylor. (1998) *Artificial Life VI: Proceedings on the Sixth International Conference on Artificial Life*.

Boston Massacheusetes: The MIT Press. [GA][AL][M]

10. Langton, Christopher G. and Katsunori Shimohara. Artificial Life V: *Proceedings of the Fifth International Conference on the Synthesis and Simulation of Living Systems*. Boston Massacheusetes: The MIT Press. [GA][AL][M]
11. Heudin, Jean-Claude. (1998) *Virtual Worlds: Synthetic Universes, Digital Life, and Complexity*. New England Complex Systems Institute Series on Complexity. Reading, Massacheusetes: Perseus Books. [AL][M]
12. Koza, John R. (1992, 1998) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, Massacheusetes: The MIT Press.[GA][M]
13. Koza, John R., Forest H. Bennett III, David Andre, and Martin A. Keane. (1999) *Genetic Programming III: Darwinian Invention and Problem Solving*. San Francisco, California: Morgan Kaufmann Publishers. [GA][AL][M]
14. Funge, David John. (1999) *AI for Games and Animation: A Cognitive Modeling Approach*. Natick, Massacheusetes, A. K. Peters. [GA][AL][G]
15. LaMothe, Andre. (1999) Tricks of the Windows Game Programming Gurus. Indianapolis, Indiana: Sams. [GA][M][G]
16. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes. (1987, 1996 2nd revised printing).*Computer Graphics Principles and Practice – 2nd Edition in C*. The Systems Programming Series. Washington, DC: Spartan Books [M][G]

Appendix A: Vocabulary

Patch – A bin of space that stores information about itself and can reference the patches it is adjacent to in the coordinate space. This idea is based upon the work of Michael Resnick and the MIT Epistemology & Learning Group's STARLOGO project.

Environment – A collection of connected patches.

Individual or "TURTLE" - An autonomous creature. A turtle contains both a local coordinate system and the methods appropriate to reference the Cartesian coordinate system. The turtle moves in three-dimensional space, and projects its location in a two-dimensional representation. The turtle concept is based in part on Ableson & Decessa Turtle Geometry from MIT.

Decision Making Turtle (DMT) – A turtle whose actions in the environment are based on the conditions it perceives and a corresponding lookup into an action table. This lookup procedure is defined through its genome.

Genome – A list of values unique to a DMT that represents which actions are to be taken in each of the conditions defined by the condition list.

Population – A collection of DMTs.

Fitness – A quantitative measure of a DMT's success relative to the goals of the simulation. One fundamental goal of the simulation was survival of the DMT, with the possibility of additional primary or secondary goals.

Mutation – Random alteration of a genome.

Crossover – Process by which two parents' genomes are used to determine the genome of offspring. Crossover preserves genetic material of the parents involved.

Reproduction – A process that selects DMTs from the population based on fitness, applies a genetic algorithm using crossover and/or mutation and then introduces the new DMTs into the environment.

Property – An instance variable that allows information to be stored within a patch or a DMT.

Condition – The perceived state of the DMT in combination with the patch it occupies and possibly its neighbors. The condition must be a member of a finite set of possible conditions, the condition list.

Condition List – The finite set of conditions derived by considering a number of properties. These properties can be external to the DMT (i.e. the amount of "food" present in the near environment), and/or internal (i.e. the amount of "energy" an individual has left).

Action – Something a DMT can do. An action is a member of a finite set of actions able to be performed by a DMT as specified within the action list.

Action List – The finite set of actions available to DMTs. In our simulations, typical actions include changing heading, moving, eating, etc.

Goal – The objective as defined by the architect of the simulation. DMTs are not inherently coded with the objective in mind, rather the reproduction of the DMTs is tailored according to a fitness scale based on a goal (or goals) to encourage the emergence of solutions.