Distributed Artificial Life toolkit

Specification

Vlad Catalin Mereuta

Distributed Artificial Life toolkit: Specification

by Vlad Catalin Mereuta

Chapter 1. The project

The problem

The field of agent based social simulation is relatively new and lacks the abundance of tools and packages available in other fields. This is exactly the problem this project will attempt to address by creating a flexible artificial life simulation environment, capable of distributing the processing requirements onto several computers.

The toolkit is to be designed to accommodate a specific class of simulations, namely the ones in which the agents are computationally intensive (neural networks, or complex processing rules). There are some such toolkits already in existence; the most prominent and well developed one is SWARM[1]. However, none of them are capable of running the simulation in a distributed manner and most of them are bound to one or two specific programming languages (Objective C and Java for SWARM). If an experimenter needed to use any other language than the one used initially for developing the toolkit a complete port would be required.

The project objectives

Brief overview

The main objective of the project is creating a toolkit which addresses the problems described above. The result should be provably more efficient on multiple machines than on a single machine, given the same set of data. The architecture should also have several levels of abstraction, allowing for different levels of customisation for particular experiments and should allow a relative language independence.

In order to achieve these objectives, the following main components will have to be integrated in the toolkit.

a map editor

This will provide the necessary functionality for creating and editing the 2-dimensional maps which provide the environment for the experiments.

a server

One or more instances of a program which keeps track of the current status of the map, the location of the agents and the interactions in between agents and their environment. The server should be able to supply all the 'sensory' inputs for any agent at any given time. It should also be able to decide whether actions that an agent would attempt to accomplish are valid or not (eg. an agent should not be allowed to move outside the map boundaries or in a location occupied by another solid object, etc). The server only holds the information *about* the agents. The agents themselves are allocated for processing purposes to one or more clients

a processing client

There will be several instances of the client program running (on one or more machines). Each client will handle the processing for one or more agents. Each agent is just the implementation of an algorithm which decides on one of the available actions based on the 'sensory inputs' (for example, the map server should be able to supply the eye-sight and smell information for an agent; based on this the agent program can decide, say, to eat some food or run away from an enemy). None of the senses or available actions are to be hard-coded - this would be limiting for the scope of the toolkit. Instead a generic and flexible framework is to be provided, which should render the task of adding these features relatively easy.

an observer

This program will use data provided by the server to render a graphical display of the artificial environment. This program should also provide support for run time data extraction and hooks for adding extra functionality.

one or more examples

These will contain simple or more complex artificial life simulations using the components above

The communication in between all these components is to be done via TCP/IP. The data transferred is encapsulated in XML. Conducting all the communication over TCP/IP should provide machine independence. Using XML for encapsulating data should provide a clear and language independent format for transferring data, a feature which should prove very important in the case of a port to a new language.

The objectives

The objectives of this project can be split into three major groups:

Core objectives

The minimum of the objectives that need to be reached in order for the project to be successful

Primary objectives

Although not vital for the success of the project, these objectives are desirable. They cannot be attained without the core objectives being completed first

Secondary objectives

These objectives shall be completed depending on the available time. None of them is vital for the success of the project.

The detailed objective list is as follows:

- Core objectives
    1. Detailed design document.
    2. Communication protocols. (specifying at least all the basic messages that can appear)
    3. Artificial Life server
        a. a threaded server which can accept connection from clients
        b. Map loading
        c. Client authentication (the server should be able to accept, acknowledge and 'use' clients)
        d. Work allocation (the server should be able to dispatch processing tasks to the clients - most likely each client will receive a pack of

   agents which will have to process. The server will keep track of where each agent is located

   e. Observer support (server should provide data necessary for the observer)

4. Client
   a. authentication support (the client should be able to login into a server)
   b. agent framework (a generic set of classes or procedures which provide the placeholder for the decision algorithm mentioned before; this is to be extended/implemented differently for each Artificial Life experiment, according to the needs of the experimenter)

5. Observer
   a. authentication support (the observer should be able to make itself known to the server and request data)
   b. basic display and recording functionality

- Primary objectives
  1. map editor for AL world maps
  2. Sample agent, able to demonstrate the functionality of the toolkit
  3. Architecture performance testing on at least three machines (given an agent which has with fake complex processing demands, the simulation should run physically quicker when more than one machine is used, thus demonstrating the utility of this project.)
  4. find processing/communication boundary (depending on the machines the program is running on, using more than one machine for a small number of simple agent can be slower, due to penalty incurred by the communication overhead). Given a sample agent and a required number of agents for the simulation, the toolkit should be able to suggest *at least* whether it is better to run the simulation on one or more machines.

- Secondary objectives
  1. Load balancing of work units in relation to processing power
  2. Client improvements and fine-tuning
  3. Complex (planning and memory) social agent for demonstration purposes
  4. Run experiment on a large network
  5. More experiments

Methods

Except for the map editor which can be created independently from the rest of the project all the goals (in the core and primary sections) are listed in the order which they should be accomplished; there is little or no room for changing the order given above. The secondary goals can be accomplished in any order.

Project type

This project is best categorised as a *development* project. The core objective of developing a flexible distributed framework for agent based artificial life simulations is a development goal. Specific algorithms and custom software will have to be developed in order to achieve this set of goals.

Timetable

Time	Objective
summer holiday; week 1,2	map editor; initial research
week 3	specification; more research and draft basic design
weeks 4-6	complete detailed specification/design document
week 7	communication protocols
weeks 8-9	initial version of the server
weeks 9-10	client
week 10	progress report
holiday	observer, module integration
week 14	complete primary objectives
week 15	finish documentation for primary and core objectives
weeks 16-17	debugging, documentation
week 18	complex agents, more experiments
week 19	compile *existing* documentation into presentation
week 20	project presentation
week 20-22	integrate latest experimental results into the report
week 22	final report

Notes

1. http://www.swarm.org

Chapter 2. Resources

Resources required

Hardware resources

Will be making use of at least two powerful (700MHz+) x86 machines. Will possibly require use of several machines in the Linux Lab toward the end of the project.

Software resources

What follows is a list of probable software resources required:

- Operating systems: Linux
- Text processing
  - DocBook XML 4.1
  - LaTeX

- Compilers/interpreters
  - GNU C++ compiler
  - SUN JDK
  - Python 2.1 interpreter (possibly)

- Development libraries (all GPL unless otherwise mentioned)
  - Threading/networking C++ library (specifics are being researched; main candidates are *Common C++* and *CoreLinux++*)
  - Xalan/Xerces C++ and Java for XML processing
  - Possibly other libraries for genetic algorithms (*galib*, *ga2*), neural networks (*inanna*)

Use of non-departmental equipment

I shall make use of non-departmental equipment in this project. In order to prevent the loss of data, the following backup strategies will be implemented:

Versioning

Both the sources for the software and the LaTeX and DocBook sources for the documentation are kept under CVS control. This enables easy recovery from mistakes. The CVS tree is kept on a different machine from the development one; this way, the most recent stable version of the work is kept on two different machines

Local backups

The CVS tree is backed up nightly on removable media, which is kept in a different place (albeit in the same residence) from the main computer.

Off-site backups

The entire CVS tree is mirrored every night into my DCS account. This is done automatically. Also, a weekly snapshot of the sources is uploaded onto the project website (hosted by Freeserve)