University of Nottingham
School of Computer Science & I.T.

Individual Dissertation
G53IDS

as part of G400
Computer Science (BSc Hons)

A HTML5 Framework for Multi-Agent Algorithms

Project Proposal

by

Ben Jenkinson
bxj08u / 4082995

October 28, 2011

1 Background and Motivation

At the moment, the visualisation of a multi-agent algorithm is often generated by and tied into the same software that is actually running the algorithm simulation.

I would like to separate the visualisation from the simulation, to open up the following possibilities:

- **Shared Resources** - Several researchers could all follow the progress of a single simulation, each from their own office, as it runs elsewhere. This would be useful for checking in on the progress of a long-running simulation.
- **Ease of Distribution** - A researcher could provide access to another collegue, though they may be geographically distant, to allow them to observe the *same* simulation and troubleshoot some erratic agent behaviour.
- **Demonstrations** - The stakeholder who commissioned the simulation to be run can view an attractive visualisation of its progress. All without the need for any additional software or installation, and without being able to alter the course of the simulation accidentally. This arrangment could be quite useful if the stakeholder is relatively non-technical and unable to run the simulation themselves.

When the visualisation client is distinct from the application or process that is running the algorithm simulation, my intention is that they would be connected by some form of internet protocol, or data-format read over the internet. The data-format would be used to describe the state of the simulation at any time, describing the agents and their properties, as well as the state of the problem environment.

This separation would mean that there is no difference between running the simulation and the client on the same computer, and running the simulation remotely. If the client has been written in HTML5 as a single-page web application, then there is no reason why the client cannot be hosted publicly as well and accessed from anywhere in the world.

This client could then effectively 'tune-in' to a stream of algorithm state-data output by a completely different machine. This would allow multiple researchers or clients to view the algorithm progress in real time while the intensive computuation is done elsewhere. Independantly, each client could then pause and inspect the data of the visualisation at any time wihtout interfering with any of the other clients.

All the while, the client could continue downloading the live stream as it is output so that the user can then skip around the timeline of the simulation from whenever they first joined the stream.

2 Aims and Objectives

The aim of this project shall be to product a framework to both animate and inspect the progress of a multi-agent algorithm.

Separation of Concerns The simulation and the visualisation components will be kept separate, leveraging a protocol/data-format that I shall define to communicate the state of the simulation. This will function over the internet, allowing the two components to be geographically distant if so required.

Usability The visualisation client shall be designed to be intuitive and attractive for the user, who may not be technically-minded.

Inspector The visualisation shall incorporate an inspector feature, allowing the playback of the algorithm to be paused and the state of the simulation as a whole, or any individual agent to be examined. This will be designed around debugging the simulation, to understand the reasoning of an agent, or the reason for some unexpected emergent behaviour. (This feature will likely assume some prior knowledge of the algorithm on the part of the user.)

Multiple Clients Ideally there will be no limit to the number of clients that could 'tune-in' to a single simulation. Allowing multiple viewers to watch, and inspect, the visualisation independantly.

3 Project Plan

3.1 Tasks

- Define a protocol, API or data-format for communicating the state of a multi-agent algorithm simulation.
  - Research types of multi-agent algorithm.
  - Research a variety of problems that can be solved by multi-agent algorithms.
  - Aquire some example problems and example algorithms to use in development of the visualisation client.
  - Define the protocol/format ensuring it would successfully describe the example problems and algorithms.
  - Write explanatory documentation for the protocol/format.

  It would be wise to limit the scope of my project to a sub-group of problems that would suit visualisation. Perhaps those areas for which an in-progress inspection of the algorithm is particularly useful.
- Investigate the potential for a server-side component for the production of the algorithm data.
  - Research existing software for the simulation of multi-agent algorithms.
  - Identify whether any of the existing software could be adapted to generate the data-format that has been defined.
  - If it is possible, adapt an existing simulation to provide the required output.
- *(Further)* Develop my own component to generate the required output stream of simulation data.
- Design the visualisation client, capable of reading a stream of the defined data-format and buffering it for playback through an ideally attractive and intuitive interface.
  - Develop the ability to read in the data-stream from a defined location.
  - Use HTML5 offline storage to buffer the data-stream for playback.
  - Develop a clear and attractive visualisation for the simulation data.
  - Develop an 'inspector' for the visualisation, whereby playback can be paused and any particular agent can be inspected.
  - *(Futher)* Draw upon my research of existing visualisation software to determine other useful features for the client to possess, then add them.

3.2 Milestones

*All dates are rough estimates, plan is subject to change. Official dates are shown in **bold***

2011	
28 Oct	**Project proposal & plan**
4 Nov	Research - Multi-agent algorithms.
11 Nov	Development - Definition and documentation of protocol/format.
25 Nov	Research - HTML5 technologies to read and store the protocol/format into the client.
28 Nov - 2 Dec	**Presentation**
16 Dec	Development - Basic client capable of reading or recieving the protocol/format.
30 Dec	Development - Client is fully capable of storing the data from the protocol/format and skipping forwards and backwards through time.
2012	
6 Jan	Research - Use of existing simulation software to generate the protocol/format/feed.
27 Jan	Development - Design and develop the visualisation of the simulation to be both attractive and intuitive.
10 Feb	Development - Adapt/develop server-side component to generate the protocol/format from the simulation.
24 Feb	**Outline & first chapter**
30 Mar	Development - System should be fully working.
7 May	**Complete dissertation**
16 May	**Demonstration**