Software Architectures for Simulations

D.P.Playne, A.P.Gerdelan and K.A.Hawick

daniel.playne@gmail.com, gerdelan@gmail.com, k.a.hawick@massey.ac.nz

Institute of Information and Mathematical Sciences, Massey University, Albany, North Shore 102-904, Auckland

Simulation-Driven Architecture

A **Simulation-Driven** architecture focuses on the simulation. In this architecture, the system is driven by the simulator (often with an iterative loop) and the entities contained within it are updated each time-step. These entities are visualised with a graphics engine to allow an observer to see the state of the simulation (see Figure: 1). This observer can control the visualiser via an input interface (keyboard, mouse etc) too change the way the graphics engine displays the entities. The operator has a degree of control over the visualisation but cannot change the simulation in any way. In this architecture the simulation and the visualisation engine have no direct knowledge of each other. The simulation operates the same way regardless of the visualisation engine and likewise the visualiser simply displays entities and is unaware that these entities are actually being updated within a simulation.

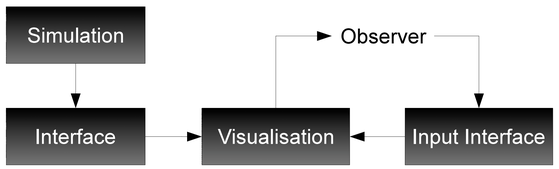

Figure 1: Simulation-Driven Architecture.

In this architecture the operation of the simulation is the point of interest. This may be in the form of a forecast where the simulation can predict what will occur given a starting configuration. Another possibility is that the visualisation may be examined by the observer to identify interesting events or phenomena occurring within the simulation. These simulators model systems such as Particles [1], Field Equations [2] and Animats [3].

Introduction

Simulations are widely employed throughout computer science for a variety of uses. A simulation is defined by a model that enforces a set of rules upon the entities within it. These entities can only operate or change their state based on the rules of the model. Simulations often require visualisation for their results to be easily interpreted. The design architecture for these simulators and visualisers is vital to the re-usability and portability of the software. Presented here are two common high-level simulation structures - **Simulation-Driven Architecture** and **Agent-Driven Architecture**.

Agent-Driven Architecture

In the **Agent-Driven** architecture the simulation provides an environment to test intelligent agents. The agents have a degree of control over one or more entities within the simulation. The decisions made by the agent change what the entity does within the simulation. In Figure: 2 the controlling agent is shown distinct from the simulation as it may be a human operator or a separate automated artificial intelligence program.

When a human operator is the controlling agent then the simulation is no longer merely visualised by the graphics engine, it is inextricably linked to it. The human operator processes the visualisation to determine the state of the simulation and from this information makes decisions about how to control its entities. This decision is then input into the simulation via an input interface (see Figure: 2).

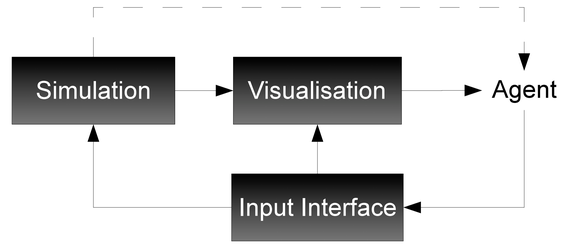

Figure 2: Agent-Driven Architecture.

The Agent-Driven architecture does not focus on the direct results from the simulation. Rather the simulation acts as an environment to test the performance of the intelligent agents controlling it. Each agent strives to achieve its goal while following the rules enforced by the simulation. Applications utilising this architecture include most modern computer games, robot soccer simulators [4] and robotic agent simulations [5].

Simulation Repeatability

Repeatability is important within Simulation-Driven architectures. Simulations that incorporate randomness within experiments should have a method of managing it. There is little point in discovering an interesting effect or phenomena if the experiment cannot be repeated. Figure: 3 shows a way of managing random-number generator (RNG) configurations. The simulator loads a configuration file, computes the simulation over a period of time and saves a final configuration. The configuration of the simulation and the RNG state is saved and can be recreated at any time. It should be noted that executing the simulation for ten time-steps, twice should produce the same final configuration as one execution for twenty time-steps.

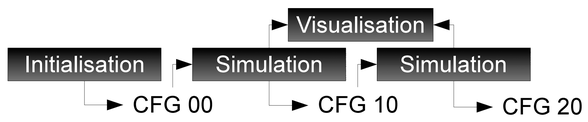

Figure 3: Configuration-Chaining Architecture.

Summary & Conclusions

As we have shown, the overall architecture can be driven from the simulation or agent perspective. A better understanding of a simulation's architecture can help to identify and avoid design-breaking features which would otherwise reduce the software's portability and re-usability.Visualisation is important in any simulation but plays different roles depending on the software goals.

In Computer Science at Albany it is possible to make use of these architectures in undergraduate and postgraduate projects (see papers: **159.333, 159.793, 159.794 and 159.795**). Recent and ongoing projects include: particle simulation, material science, animat agents, robotic tank control (see background image) and network simulations. We believe these architecture concepts are applicable to simulations across many disciplines.

References

[1] D. P. Playne, "Notes on particle simulation and visualisation," Hons. Thesis, Computer Science, Massey University, June 2008.

[2] K. A. Hawick and D. P. Playne, "Modelling and visualizing the cahn-hilliard-cook equation," in *Proc. 2008 International Conference on Modeling, Simulation and Visualization Methods (MSV'08)*, July 2008.

[3] H. A. James, C. J. Scogings, and K. A. Hawick, "A framework and simulation engine for studying artificial life," *Research Letters in the Information and Mathematical Sciences*, vol. 6, pp. 143–155, May 2004.

[4] A. Gerdelan and N. Reyes, "A Novel Hybrid Fuzzy A* Robot Navigation System for Target Pursuit and Obstacle Avoidance," in *Proceedings of the First Korean-New Zealand Joint Workshop on Advance of Computational Intelligence Methods and Applications*, vol. vol.1, (Auckland, New Zealand), pp. pp. 75–79, 2006.

[5] A. P. Gerdelan, "The Mark IV 3D Battlefield Simulation Engine." http://markiv.sf.net, 2008.