Chapter 2: Related Work

A few things can be said in general about agent architectures designed for real-time execution in dynamic environments. Because agent architectures must act quickly with partial or inaccurate knowledge of the world, none of them necessarily produces optimal solutions to a problem. Instead, they are designed to produce "good enough" actions quickly and to change plans whenever necessary. To achieve the requisite speed, almost all have a reactive component that relies heavily on precomputed mappings of situations onto appropriate actions. The remainder of this chapter discusses agent architectures that have been used or could be used to produce characters in interactive story systems. It focuses on these architecture's advantages and limitations.

2.1 Agent Network Architecture

Maes [Maes, 1989; Maes, 1991] describes a hybrid architecture called an "Agent Network Architecture" (ANA), in which the selection of a particular behavior is an emergent property of a parallel process. Her architecture was the starting point for the PHISH-Nets architecture, so there are many similarities between them.

ANA was inspired by Minsky's *Society of Mind* theory [Minsky, 1986], which states that the mind is made up of a society of interacting parts. Each of these parts, on its own, is mindless, but intelligent behavior emerges from the interaction among these parts. In an agent network, a character is described in terms of the actions that character can perform. These actions are low-level behaviors such as "walk towards food" and do not on their own exhibit intelligent behavior. Actions are selected by a spreading activation mechanism through a network where each action is a node. These nodes are linked together at compile time with links that represent causal relations among the actions in a character's repertoire. As an agent operates in the world, so-called "activation energy" comes into the network from both internal goals and from the state of the world. This energy is then spread through the network by a process that makes activation energy accumulate in the most relevant behaviors. When an executable behavior acquires activation energy beyond a certain threshold, it is enacted.

The architecture has been shown to be fast, robust, and flexible in a dynamic and unpredictable environment, and it can also handle multiple goals gracefully [Maes, 1991]. ANA networks are also compiled from a relatively straightforward description of actions, and are thus both easy to create and to extend. However, they do not support parameterized actions such as *Walk-to <location>*, and as a consequence have trouble acting in environments where there are complex interactions among possible behaviors. They can also get stuck in loops where the same action is performed over and over, especially if there are no methods to satisfy an action's preconditions, or if all such methods have failed.

Tyrrell [Tyrrell, 1994] also points out several shortcomings in ANA. Most notably, he points out that the architecture favors appetitive (preparatory) actions over consummatory (achieving) actions. He also demonstrates a flaw in division of energy among behavior-nodes that causes actions that accomplish many goals to be starved out. A similar flaw penalizes actions whose preconditions can be solved in many ways. These last two flaws will be discussed in more detail in Section 3.2.3.

The PHISH-Nets architecture is inspired by ANA, but addresses the above shortcomings so it can be used to create virtual characters for interactive story-systems.

2.2 Hamsterdam

The Hamsterdam behavior system [Blumberg, 1995] is similar to ANA in that activation energy is passed between nodes representing behaviors, but these nodes are arranged in a hierarchy rather than in a flat network. The algorithm is based strongly on ethology[1], and is therefore extremely well suited to the creation of virtual animals. The Hamsterdam architecture is used in the ALIVE [Maes et al., 1995] project, which situates a human in a virtual 3D environment with animated characters, and has also been used in the interactive story "Dogmatic" [Galyean, 1995].

Hamsterdam was designed to operate in an uncertain world with incomplete knowledge, handle multiple goals, and to act appropriately when actions do not have the desired effect. It is also designed to work in a continuous world where actions are not atomic and take time to accomplish. However, to add a new behavior an author must hand-code different situations in which the action can be used. Furthermore, unlike ANA, Hamsterdam is primarily a reactive system, and is not designed to handle complex sequences of actions to achieve a goal. This characteristic makes it a less attractive architecture for designing characters that must engage in sophisticated planning-style tasks.

Blumberg's Hamsterdam system also implements a sophisticated motor system for controlling graphical characters that incorporates blending of simultaneous actions, animation of focus-of-attention cues, and incorporation of emotional state into the performance of actions. This motor system is being used to implement the graphics and low-level actions of characters in the test runs of this thesis, even though the Hamsterdam behavior system is not used.

1. The study of animal behavior

2.3 Hap

The Hap reactive agent architecture [Loyall, 1991] is a part of the Oz project at CMU and was designed specifically to create characters in interactive story environments. Hap creates an "active plan tree" which consists of a sequence of actions necessary to achieve a goal. Each time step, the active plan tree is first pruned of all plans that are nonsensical and goals that have been spontaneously fulfilled. Then one of the goals from the tree is chosen to be executed, on the basis of hand-coded goal priorities. Finally, if the chosen goal is a primitive action, it is executed; otherwise subgoals are indexed from a plan memory and placed into the active plan tree for subsequent processing. The HAP architecture has been modified to provide real-time control of three graphical characters, called *woggles*, in an animated world titled *Edge of Intention* [Loyall, 1993]. This real-time system can handle continuous spaces, and also blends actions together such that the animation for an action changes depending on what action is coming next. It also incorporates a model of emotion that affects both the actions a character takes and the manner in which those actions are performed.

Hap has several of the features described in Section 1.3. The architecture is designed to be opportunistic, handle multiple goals, specify sequential or parallel plans for accomplishing a goal, and to act appropriately when actions do not have the desired effect. It is also designed such that new plans can be easily added without affecting existing plans. However, there are a few limitations to the architecture, especially with respect to goal arbitration. Hap uses hand-coded priorities to arbitrate between multiple top-level goals, but does not take into account how actions may affect less important goals. Opportunistic behavior is also limited to taking advantage of situations where a high-priority goal is spontaneously fulfilled; Hap will not take advantage when a lower-priority subgoal is satisfied. For example, if a character came across a water fountain when it was more hungry than thirsty, it would still look for food rather than drink. These are some of the limitations that the PHISH-Nets algorithm is designed to overcome.

2.4 The Virtual Theater Project

Stanford's Virtual Theater Project has applied the BB1 blackboard architecture [Hayes-Roth, 1995] to creating virtual characters for interactive story systems. Unlike HAP, ANA, or the PHISH-Nets architecture, this architecture's primary focus is the production of actions that are in character and appropriate for a character's mood, rather than producing actions that achieve goals. Interaction with a character is structured around the model of a virtual stage, in which a human director gives general guidelines to a character in terms of a loose script. For example, a character might be given the script to "Stand at the wall for five seconds, then play until a certain event happens, then sit down" [Hayes-Roth, 1996]. These scripts can either be written out in advance (virtual actors) or produced in real-time by a human director (virtual puppets). Characters take the general human-provided instruction such as "play," and within those constraints act in character, e.g., by either playing energetically or quietly. The system has a simple model of social interaction that characters use to infer the mental state of other characters. It also has a simple emotional model that is used to color a character's actions.

2.5 Real Time Responsive Animation

Ken Perlin at NYU has developed what he calls "real time graphic puppets." [Perlin, 1995] These animated characters can perform hand-coded actions while moving in a life-like manner. However, the characters do not encode any real intentionality: they only encode a visual impression of personality and life-like behavior. Although this work could provide an alternate implementation for a character's low-level motor functionality, it does not provide an action selection mechanism beyond a low-level scripting system.

2.6 Terzopolous' Artificial Fishes

Demetri Terzopoulos at the University of Toronto has developed a physically based simulation of artificial fishes [Terzopolous, 1995]. His work concentrates primarily on realistic modeling of fish motion and bodies, but he has also developed a simple architecture to produce fish-like behavior. His action selection algorithm uses a few very simple hierarchical rules to choose between basic intensions such as *avoid*, *escape*, and *mate*, based on parameters such as *fear* and *libido*. While this behavior system works well for producing fish-like behavior, it does not scale well to handling sophisticated sequences of actions where multiple actions can accomplish a single goal.