BAMUVA: A Robot Behaviour Based Architecture to Create Virtual Creatures

Carlos Delgado, Ruth Aylett

Centre for Virtual Environments
The University of Salford
Business House
University Road
Salford, M5 4WT

Tel: 0161 295 2918
Fax: 0161 295 2925
e-mail: C.Delgado@pgr.salford.ac.uk
             R.S.Aylett@salford.ac.uk

Abstract. In this paper we describe BAMUVA. BAMUVA is Robot based behaviourual architecture designed to create Intelligent Agents in a Virtual Environment, where each agent senses obstacles and other moving creatures in the environment and reacts to them. The architecture consists of a Behaviour layer, a motor layer and a library of sensors.

Keywords: Agents, Virtual Environments, Behaviour robotics, CAVE, Maverik, Linux, OpenGL, Animation.

1. Introduction.

This work is inspired by behaviour based architectures in robots, (Brooks, 1986)(Arkin, 1999). One of the pioneers in the field was Braitenberg (Braitenberg, 1984). He produced architectures in which sensors are directly coupled to motors in order to transmit inhibitory and excitatory influences. He created a wide range of vehicles, producing sophisticated emergent behaviour, which he labelled with terms such as cowardice, aggression, and even love. However these systems were inflexible, custom machines and were not re-programmable.
In the mid 1980s the subsumption mechanism was developed (Brooks, 1986), where task-achieving behaviours are represented as separate layers. Individual layers work on individual goals concurrently and asynchronously. At the lowest level each behaviour is represented using an augmented finite state machine model, and higher levels are allowed to subsume the activity of the lower ones. Again inhibitor and excitatory connections allow complex emergent behaviour to be produced by time-slicing control of the robot between different internal behaviours.

An advantage of these systems is that they are inherently modular from a software design perspective. This enables a reactive robotic system designer to expand his robot competency by adding new behaviours without redesigning or discarding the old ones.

This work has become very relevant to the field of Virtual Environments, which has been developing steadily, with cheaper devices and growing expertise; there are new near photo realistic environments. One of the challenges that the research community is trying to meet is to populate the environments with believable virtual creatures.

Several approaches have been developed to address this issue, with two main streams. One is the bottom-up approach, where complex behaviour emerge from the interaction of simple internal behaviours, as in Creatures, or Silas (Blumberg, 1996). On the other extreme, the top-down approach, the programmer explicitly defines the behaviour of the creature. However this approach does not scale up well, especially in the case of multiple creatures interacting with each other.

We argue that to create a Virtual Creature we need perception of the environment through virtual sensors, “Intelligence” to tell the creature how it will react to the changes in the environment state, and a motor control to move the creature in the virtual environment.

2. BAMUVA

Our goal is to create an architecture to support Multiple Creatures in a Virtual Environment, where complex behaviour can emerge from interactions between agents and each agent with the environment. To achieve this we are using a Robot based behavioural architecture (Barnes, 1997), where each agent senses obstacles and other moving creatures in the environment and reacts to them.

The architecture was modelled using UML, because we want to extend the architecture and to make it possible to use it with real robots and virtual creatures, where the only thing that needs to be changed is the actual implementation of the Creature, how it senses and acts in its environment.

The architecture is divided into three parts.
2.1 Behaviour Based Architecture for Multiple Virtual Agents.

The basic component in the architecture is the behaviour pattern, \( \text{bp} \); see Figure 1. A \( \text{bp} \) defines a desired motion response, which is a function of a given sensory stimulus. Associated with every response is a measure of its utility or importance. Hence a \( \text{bp} \) defines not only what the motion response should be for a given sensor input, but it also provides a measure as how the relative importance of this response varies with respect to the same sensor input.

![Figure 1. Behaviour Class Diagram.](image)

\( \text{Bp} \)'s are collected into behaviour packets: a collection of patterns, which linked together can accomplish a goal. For example through a set of patterns like object-avoidance, navigate-towards-beacon and wander, the creature can accomplish the task of reaching a beacon in a cluttered environment.

The next abstraction is the behaviour script, which is a list of packets each of which is triggered by a sensory pre-condition A behaviour script, by permitting behaviour packets to be activated and deactivated, supports goal searching, allowing a behaviourally driven creature to accomplish tasks with complex sequential structure.

The architecture is an Object Oriented extension to the Behaviour Synthesis Architecture, which was developed in the University of Salford (Barnes, 1997), to accomplish a task through cooperating robots. This work used ethological knowledge (McFarland, 1999]). While each robot had a repertoire of simple behaviour patterns, complexity emerged through interactions between behaviour patterns and between robots.
2.2 Motor Layer.

The motor layer which is used to store movements, i.e. walking, running, reaching, etc., for each kind of creature (Delgado, 2000). This is done by selecting key joints in the desired animal (Gray, 1968). In our case an ostrich and a kangaroo (Muybridge, 1887). With motion captured we then digitised each joint, through the number of desired frames. Later on we used data to compute the angle of each joint in relation to its parent.

The Virtual Environment is made from static objects, moving objects and creatures (moving objects with “brains”). The abstraction of these classes is shown in figure 2.

![Figure 2. Behaviour class diagram](image)

The Brain of the Creature senses the environment and sends a response, which is converted by the Motor Layer into animation. Each creature has a library of movements, and each creature has a state recording which movement it is performing.

2.3 Sensor Library

The “brain” in the creature senses the environment and then through a synthesis mechanism, reconciles conflicting behaviours. The Behaviour Architecture generates a response, which is passed to the motor layer, where through forward kinematics the creature is animated, see figure 3.

In the example implementation, the sensors consist of a distance sensor and a hunger sensor. There is also sensor abstraction, so that the user of the architecture has a common interface for the sensors and could expand the library of sensors to include others.
3. Sample Application.

The program runs on the CAVE (Cruz-Nera, 1993), which is connected to a Onyx 2 with 14 R10K processors and 5 Infinity Reality graphics pipeline, in the Centre for Virtual Environments, but it also runs on a SGI Workstation, and on a Linux workstation with a Voodoo 3000 video card.

There are four windows displayed, see figure 4. The views were created using a camera for the front window, and then created the other three windows using simple matrix multiplications (Foley, 1992). To render the environment we used Maverik API (Hubbold, 1996) a virtual environment interface kernel, developed at the University of Manchester.

4. Conclusions and Future Work.

We have obtained interesting results, for example a flocking behaviour (Reynolds, 1986) has emerged with several creatures, each with its own behaviour packets containing Wander and avoid-obstacles behaviour patterns.
The system has been implemented in an immerse environment as in the CAVE, although it can also run on a low-end computer such as a Linux workstation. The modularity with which the architecture was designed made it easy to incorporate new behaviours.

In the future we want to port the Virtual Environment to Performer because Maverik has had some problems with the Onyx machine. Though the performance in Linux is very good.

We want to develop or use a better motor controller to make the creature movements more fluid, like the actors in IMPROV System (Perlin, 1996).

5. Acknowledgments.

Thanks to CDF, AgroIndustrias CEJA, Universidad Bonaterra and The University of Salford for their support.

6. References


BAMUVA: A Robot Behaviour Based Architecture to Create Virtual Creatures.


Hubbold, R., MAVERIK the Manchester Virtual Environment Interface Kernel. 3rd Eurographics Workshop on Virtual Environments, 1996


Muybridge, E. Animals in Motion, 1887


