An Unreal Based Platform for Developing Intelligent Virtual Agents

N. AVRADINIS, S. VOSINAKIS, T. PANAYIOTOPOULOS,
A. BELESIOTIS, I. GIANNAKAS, R. KOUTSIAMANIS, K. TILELIS
Knowledge Engineering Lab, Department of Informatics
University of Piraeus
80 Karaoli & Dimitriou str., Piraeus 18534
GREECE
avrad@unipi.gr, themisp@unipi.gr

Abstract: - In this paper we suggest an approach for developing programmable intelligent virtual agents over Unreal. We propose various techniques for manipulating creating and modifying Unreal Engine's actors, as well as a method for developing an additional external controller responsible for intelligent decision making by creating programmable agents.

Key-Words: - Virtual Agents, Virtual Environments, Task Definition, Planning, Simulation, Programmable Agents

1 Introduction

Synthetic characters alongside with virtual environments can be used in dynamic simulations in order to increase the simulation's believability. Such agents should not only be believable but also be capable of interacting with their environment and other agents, [1,2,4], and simple to program for accomplishing numerous tasks. The virtual environment should aid the agents' roles and should be able to adapt to different types of simulations [3].

Previous research efforts in the Knowledge Engineer Laboratory has shown that it is possible to develop Intelligent Virtual Agents capable of exhibiting complex behaviors, either over the network in a distributed architecture, [1,10], or over the web, [3], or as a stand-alone application, [5,7,11]. On the other hand, the Unreal Engine has been used for interactive storytelling, [9].

In this paper we propose a framework for developing programmable agents, using the Unreal Engine for embodying the agents and visualizing the environment.

The Unreal Engine fulfils most of the prerequisites stated. It is a freely available, up-to-date 3D graphics engine, providing character animation and basic interaction. Moreover, via the UnrealScript scripting language the engine can be parameterized and extended to real-time programmed scenarios.

This paper is structured as follows. In Section 2 we demonstrate a generic IVA architecture. In section 3 we present an overview of the agent's architecture in Unreal and we discuss the use of UnrealScript. In section 4, we discuss the possible solutions for developing programmable agents over Unreal and we propose our approach. In section 5 we demonstrate an example of a dynamic simulation. Finally, in section 6, conclusions along with future work are discussed.

2 A Generic IVA Architecture

For our platform we adopted an IVA architecture developed by our Laboratory, which is already presented in [8]. The architecture is shown in Figure1.

An IVA has the following characteristics which define its behaviour:

- High level beliefs
- Ground beliefs
- Basic emotions
- Object relations
- Geometry data
- Attributes
- High level drives
- Personality

The agent's decision mechanism consists of three layers:

- The cognitive layer
- The non-cognitive layer
- The low level layer

In the cognitive layer, the High level Beliefs, Basic emotions, object relations, attributes and high level drives dictate the agent's causality and appraisal attribution, thus forming a decision. Based on that decision, conscious actions are defined. The results of each step of the process are evaluated and alter the agent's appraisal and causality attribution as well as their characteristics accordingly.

In the non-cognitive layer the ground beliefs, basic emotions, geometry data, the attributes and the personality invoke reactive actions. That means that

the appraisal and causality attribution stage is omitted, thus resulting in the agents performing reflexive actions.

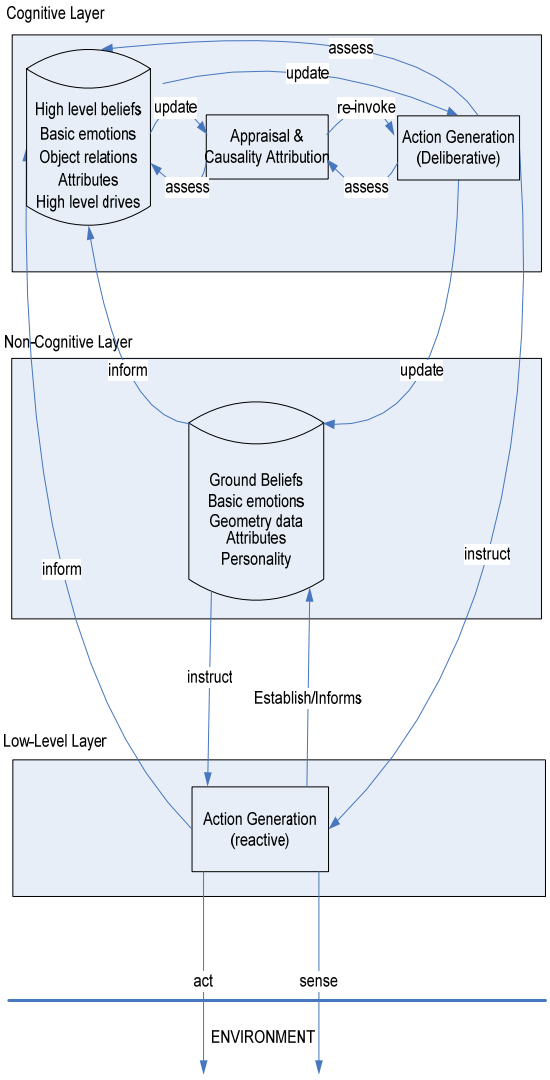

Fig.1 Generic IVA Architecture

Lastly, in the low level layer, reactive actions are generated with no consideration of the IVA's characteristics. They are the agent's reflexes. The aforementioned actions update the main characteristics.

The action taking mechanism is activated by the sensory stimuli generated in the environment. The actions which are performed affect the environment.

3 Unreal Actors & Architecture

The Unreal Engine is a general purpose game engine created by Epic Inc. It consists of a freely available runtime module that can be used to implement and visualize 3D virtual environments, as well as a development platform for map design and script editing.

The Unreal engine is fully controllable and extendable by script code written in a proprietary language, UnrealScript. UnrealScript is an Object Oriented Programming (OOP) Language with C++ like semantics and syntax offering all the facilities of OOP languages, like classes and inheritance, but also implementing a simple state machine and basic event-driven programming.

It is used to describe the characteristics, the behavior, the interactions and the appearance of the virtual environment and its components providing the developers with a powerful language capable of manipulating the Unreal Engine.

We have tried to depict the behavioral model of Unreal Actors as a sense-decide-act loop, which is achieved by appropriate sensors and effectors. The real architecture is not known to us in detail, as the engine it is undocumented, except to the open source UnrealScript part and a few informal tutorials on the web. Nevertheless, the proposed approach seems to fit well to the Engine operation and can be used to our own research purposes.

Figure2 depicts the architecture of the virtual agent's behavioral model. The following definitions provide an understanding of the terms used.

- An event and sensory stimulus is a cause for making a decision. Events are triggered by actions of other agents or the "laws" of the world.
- A sensor is the means through which the agent receives the messages of the environment.
- The state machine is a set of states and the rules describing the transitions between them.
- Effectors are the means through which the agent-decided actions are applied to the environment or other agents.

The agent perceives the environment through the basic sensors and events provided by the engine. The messages created are received by the senses of the agent. For example, a bump on another agent creates an event triggering a sensor, whereas a message notifying the agent of the position of a specific object inside their field of view is a visual stimulus. Following the reception of the messages, depending on the current state, a decision for an action is made. The state machine along with the incoming messages defines that decision.

The actions are categorized as follows:

- State changing
- Attribute changing

- Sense activating
- Performing

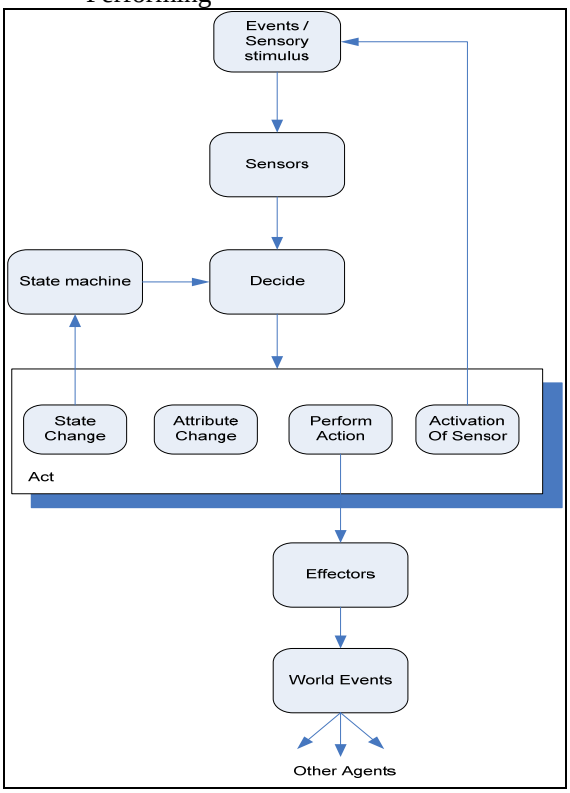

Fig.2 Unreal Agent's Architecture

The case of state changing is self explanatory, for example the transition between idling and wandering state. Actions can also result in the altering of an agent's attributes, which can be best described as "properties", for example their location. An agent can activate one or more senses, for example, vision. Finally an agent undertakes an action causing a world event, subsequently triggering the sensory mechanisms of other agents [4].

In general the decision making process of Intelligent Virtual Agents can be categorized as follows [8]

- Low level behavior
- Non cognitive – affective decision making
- High level decision making

The low level behavior takes into account only the information that originates from the sensors and produces reflexive reactions, whereas the non cognitive decision making also receives input from the agent's affective state and produces more "intelligent" reactions. The high level decision making takes into account agent beliefs, performs reasoning and planning and in many cases implements a BDI, i.e. Belief-Desire-Intention, architecture.

In Unreal, the low level and a small part of non-cognitive processes has been implemented. There are no affective states and high level decision making is not implemented.

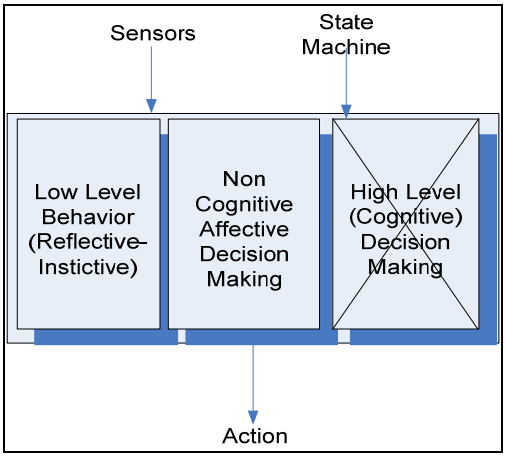

Fig.3. General purpose decision mechanism and the case of Unreal

4 Developing Programmable Agents

Our main goal is to create an agent, whose behavior can be programmed using AI techniques, thus a programmable intelligent virtual agent, following the example of [10,11]

4.1 An UnrealScript Oriented Approach

A first approach suggests implementing the agent using UnrealScript, either by modifying an already existing agent, or creating a new one from scratch.

- When modifying an agent, the developer should at first comprehend the underlying code of the existing agent. The next step consists of finding the code segments that are subject to change and modifying them accordingly without breaking the existing code's consistency.
- When creating a new agent, the developer should create a low level framework, describing basic agent components such as animations, physics and texturing, and afterwards determine the functionality of the agent and proceed with the implementation.

The aforementioned solutions would have been simple had it not been for:

- Insufficient documentation
- The existing agents' combat-oriented behavior

- Increased code complexity and size
- Non readable/modifiable code in the engine
- Lack of utility libraries (e.g. I/O, string manipulation)
- Lack of direct connectivity with high level languages used in AI programming (e.g. Prolog, Lisp)

4.2 An External Controller Approach

The suggested solution is to use a language other than UnrealScript to implement the agent's behavior. The basic concept of such an approach is to utilize the sensors / effectors of Unreal Actors, i.e. to take advantage of the unreal environment in order to receive messages, as well as to visualize the world. The decision mechanism of the unreal engine is bypassed, and is handed over to a new application, the external controller, inspired by the case of DIVA, [1]. This application can be enhanced via a high level language such as Prolog, benefiting from its numerous advantages.

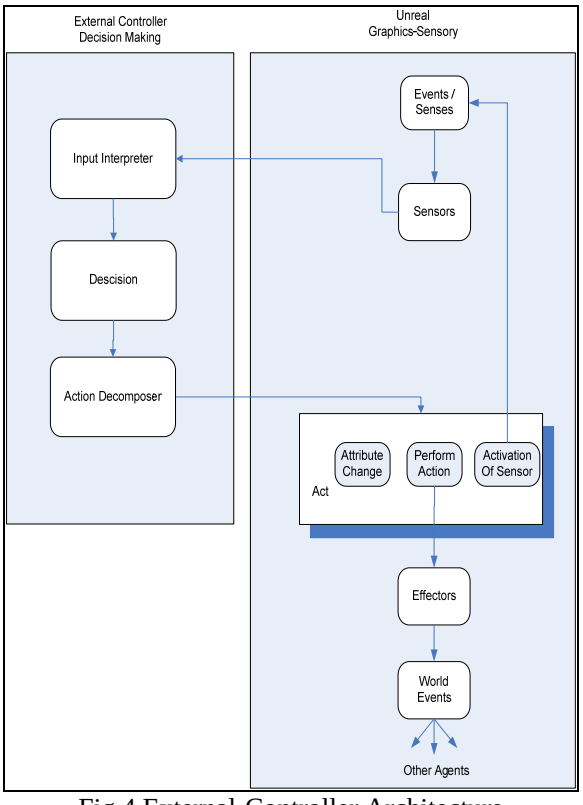

Fig.4 External-Controller Architecture

In the unreal part, a framework to handle the agent and its sensory input is implemented. When a message of the environment is captured by the agent's sensors, the external application is notified. Information about the type, the time-stamp, as well as the message itself is passed over. The unreal part is also capable of receiving messages from the external controller, which are translated into certain actions.

The actions of the agent have been broken down to two main categories, complex and elementary. The complex actions consist of a series of other actions, whereas the elementary, when combined, form a complex action. For example moving to a specific position is an elementary action, whilst following a moving object is complex action, as it can be broken down to several elementary actions, i.e. moving to different specific positions.

In the external controller, the behavior of the agent is defined. Based on the incoming messages, decisions supporting the agents' goals are made and actions are undertaken. These are in most cases high-level and complex and should be analyzed to a set of elementary actions which are transmitted to unreal and realized.

The external controller in coordination with the modified unreal engine provides a framework for developing programmable agents. The behavior of agents can be shaped, depending on the concept of a given simulation.

5 Illustrative Example

In order to demonstrate the proposed framework, a simulation has been created. The scenario of the simulation is the usage of predefined ingredients found in a kitchen, as instructed in a recipe.

Fig.6 Simulation, Agent Searching For Ingredients

The Unreal Engine displays the virtual environment, i.e. a kitchen, the agents, i.e. the cook, handles the senses of the agent and stores information concerning the objects in use, i.e. the objects name. The agent senses their environment, via its sensors. For example it is capable of viewing the objects and their attributes. The information

received by the agent's senses, in an appropriate format, is transmitted to the external controller.

The external controller functions like the "brain" of the agent. It withholds information concerning the recipe, such as the ingredients used, their order and the way they are used. According to the recipe the agent is guided through the cooking.

6 Conclusions & Future Work

We have presented a framework for the development of programmable agents and their use in dynamic simulations over the Unreal Engine.

We are currently trying to apply the framework to a network-based, multiprocessing, distributed environment, scaling our architecture. In addition, we intend to develop more believable and intelligent agents and employ their use in more demanding scenarios, [5]. Such scenarios can be behavioral control in emergency situations or interactive storytelling [9].

References:

[1] S.Vosinakis,G.Anastassakis, T. Panayiotopoulos, DIVA: Distributed Intelligent Virtual Agents, *Workshop on Intelligent Virtual Agents, Virtual Agents 99*, The Centre for Virtual Environments, University of Salford, 1999, pp. 131-134

[2] Michael Wooldridge, *An Introduction to Multi-agent Systems*, John Wiley and Sons Ltd, 2002

[3] T. Panayiotopoulos, S. Vosinakis, J. Kalligatsis, K. Kabassi, Web-Based, Dynamic and Intelligent Simulation Systems, *Intelligent Systems and Control International Conference* (ISC 2000), Honolulu, Hawaii, USA, August 14-16, 2000, pp. 398-403.

[4] Gerhard Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press, 2000

[5] V.S.Belessiotis, S.Vosinakis, T.Panayiotopoulos, The use of the Virtual Agent SimHuman in the ISM scenario system, *Advances in Automation, Multimedia and Video Systems, and Modern Computer Science*, V.V. Kluev, C.E.D'Attellis, N.E. Mastorakis (Eds.), Electrical and Computer Engineering Series, WSES Press, 2001, pp.97-101.

[6] Spyros Vosinakis, Themis Panayiotopoulos, A Task Definition Language for Virtual Agents, *Journal of WSCG*, Vol.11, No.3., UNION Agency, 2003, pp. 512-519.

[7] Spyros Vosinakis, Themis Panayiotopoulos, Programmable Agent Perception in Intelligent Virtual Environments, *Lecture Notes in Artificial Intelligence*, Vol.2792, Springer, 2003, pp.202-206.

[8] Nikos Avradinis, Spyros Vosinakis, Themis Panayiotopoulos, Synthetic Characters with Emotional States, *Lecture Notes in Artificial Intelligence*, Vol.3025, Springer, 2004, pp.505-514.

[9] Nikos Avradinis, Themis Panayiotopoulos, Ruth Aylett, Continuous Planning for Virtual environments, *Intelligent Techniques for Planning*, I. Vlachavas, (Ed), Idea Group Publishing, in press, 2004

[10] George Anastassakis, Themis Panayiotopoulos, A System for Logic based Intelligent Virtual Agents, *International Journal On Artificial Intelligence Tools*, in press**,** 2004

[11] Spyros Vosinakis, Themis Panayiotopoulos, A tool for constructing 3D environments with Virtual Agents, *Multimedia Tools and Applications Journal*, in press, 2004.