Utilizing HLA for Connecting Virtual Worlds to Pogamut

Tomáš Plch (tomas.plch@gmail.com), Tomáš Jedlička (jedlickat@gmail.com), Cyril Brom (brom@ksvi.mff.cuni.cz)
Department of Software and Computer Science Education
Faculty of Mathematics and Physics, Charles University in Prague
Malostranské nám. 25, Prague, 11800 Prague, Czech republic

Abstract. *The research in the field of human-like agents, is limited by the lack of accessible, large, dynamic, and real-time 3D worlds. The fast AI prototyping tool Pogamut makes some of these environments, like the Unreal Tournament 2004 computer game, accessible. However, multitude of these computer game worlds is limited or not suitable for large scale simulations. We aspire to implement an architecture able to connect different 3D worlds via the High Level Architecture (HLA) to Pogamut, creating a sophisticated platform, called HLA Proxy, for making large scale simulations accessible. The HLA represents a standard for interconnecting different simulation environments. Here, we present our effort to connect the Virtual Battle Space 2 military training platform to the Pogamut fast prototyping tool via HLA. HLA Proxy is currently in development, but the first working prototype is scheduled to be done in September 2010.*

1 Introduction

The field of human-like agent research currently lacks a large scale easily accessible simulation environment. In most cases, researchers either create their own simulation environment or try to utilize ("hack into") an existing engine – in most cases these are computer game engines like Unreal Engine [1], Source Engine [2] etc.

There is also need for a developing and testing ground for developers creating agents for computer simulations (e.g. for military personnel training [3]). Developers and designers need a place which they can exploit by creating prototypes and experimenting with ideas.

This gap was recently (about 2-3 years ago) filled with the fast prototyping tool called Pogamut [4], which is being continuously developed at our laboratory. Pogamut, being a plugin for the NetBeans IDE, provides easy access into the Unreal Tournament 2004 (UT2004) [5] video game environment. Unfortunately UT2004 is not suitable for larger simulations incorporating higher number (10+) of agents. This is largely due to architectural limitations of UT2004.

We identified the Virtual Battle Space 2 (VBS2) [6] military simulation platform as a suitable candidate host engine for large scale simulations. VBS2 provides a large scale virtual environment able to handle large numbers of individual agents (>100). We also want to introduce the High Level Architecture (HLA), being the industry standard [7], [9] for interconnecting simulation environments, into Pogamut, making the platform more versatile and capable.

The idea of HLA is based on two principles – the Run Time Interface (RTI) and the Federation Object Model. The RTI provides the communication means for simulation participants (called *federates*) and the FOM provides data structure description and interaction (e.g. grenade explosion) description for the actual interaction of federates.

The HLA can be viewed as a distributed database, where federates can subscribe to receive attribute updates, acquire ownership of objects and fire interactions described in the FOM.

2 HLA Proxy Architecture

Our basic idea is to create a middleware capable of translating Pogamut's requests on data, as well as action execution into the HLA environment (Figure 1).

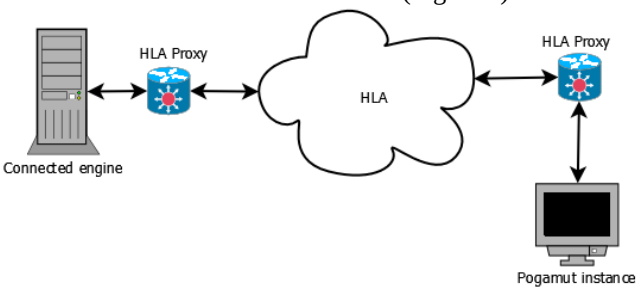

Figure 1: HLA Proxy per Pogamut instance

Secondly, we intend to create a more complex mechanism, providing a data engine (within the HLA Proxy), where inferences on data, as well as creating higher data abstractions, can be performed and presented. Thirdly, we intended to extend the capability of the Pogamut platform by providing a communication node for multiple instances of Pogamut.

We are developing our own Artificial Agents FOM (AA FOM) related to human-like agents. The AA FOM differs from the currently employed military or commonly available FOMs. Our main goal for the first version of AA FOM is to develop a minimalistic universal description of the world and to implement a management of object classes for remote control of the engine's basic support systems, like map loading, simulation termination, utility and monitoring functions etc.

Our architecture's design aims at the intended evolution of our FOM, to make it usable in future versions of HLA Proxy with minimal source code changes or limitations to the FOM design.

Currently, we finalize development of HLA Proxy's core for plugin management and implement the capability to reflect custom FOMs utilizing C++ templates.

2.1 Modular architecture

The HLA Proxy project is designed as a modular architecture. This modularity is needed to support various network topologies and usage scenarios. Multiple engines do not share the same approach for adding new features. Therefore our architecture must be flexible, to be easily deployed under various conditions.

HLA Proxy architecture consists of three elements – the *Plugin Manager*, *plugins*, and *components.*

2.1.1 Plugin Manager

The Plugin Manager provides necessary methods for loading plugins and obtaining components stored inside plugins. It also supports resolving dependencies amongst plugins at load time. Developers using the HLA Proxy middleware must instantiate the Plugin Manager before using anything from HLA Proxy's plugins.

2.1.2 Plugins

Plugins are standard shared libraries (e.g. a "dll" file in the Windows operating system), which are loaded by a Plugin Manager instance. The main purpose of a plugin is to provide a container for various components.

2.1.3 Components

Components in the HLA Proxy architecture implement the *HLA Proxy component interface* and perform computational work. They may have minimalistic form as a simple class in the C++ programming language. On the other hand components may be composed of multiple classes. In this case, there has to be one class acting as the component interface implementation.

2.2 Shadow copy

One of the project's goals is to support multiple threads accessing data components (representing AA FOM's objects) and attributes within. The HLA Proxy itself is designed as a threaded application.

It is not possible to implement classic simple synchronization using standard synchronization primitives and not lose performance. We also wanted to avoid transactions to keep the use of our middleware simple.

We distinguish two cases of modifying component's attributes (referred as *object* in the rest of this section).

2.2.1 Reading

When a developer tries to read attribute values, we are forced to lock the object to provide data integrity and consistency during the operation or calculation performed on based on the data. There is no limit on how long such lock can be in place and can result in blocking other federates.

To solve this situation, we freeze the object until all callbacks finish and the locks are released. When the developer tries to lock the object, he obtains a real lock on a standard synchronization primitive, but only for a short time period. After locking, the object creates a copy of all registered attributes and unlocks the standard synchronization lock. After this, the developer works with copy of data, which does not change in time, until he unlocks the object.

2.2.2 Writing

Updating attributes with new values takes only short amounts of time and thanks to the HLA standard, no writing collisions can arise, due to attribute ownership.

3 Future work

We intend to finish our prototype implementation of the HLA Proxy and connect it to VBS2. We also intend to integrate the prototype into Pogamut. Our second goal is to create the AA FOM based on our experience with artificial agents intended for the use in military simulations and computer games.

Later on, we intend, based on the collected data from the connected VBS2 simulation, to create a TCP/IP protocol between the HLA Proxy standalone gateway and Pogamut.

Finally, we want to integrate the data inference capability and data processing features into the HLA Proxy to provide Pogamut with higher data abstractions the Intelligent Virtual Agents can work with.

4 Conclusion

We presented our architecture designed for connecting our AI prototyping tool Pogamut, to the VBS2 or any other HLA capable environment. We presented our versatile architecture, where plugins and components can be used to represent the HLA world simply but effectively as well as provide additional services. The architecture is aimed at performance, modularity, and adaptability.

We believe the architecture can extend Pogamut (and other tools) to utilize the HLA and provide a wider range of virtual environment available for AI research and AI related development.

Acknowledgement

Writing of this paper was partially supported by the project CZ.2.17/3.1.00/31162 that is financed by the European Social Fund and the Budget of the Municipality of Prague. The research related to HLAProxy was also supported by a student grant GA UK No. 0449/2010 /A-INF/MFF, and by project P103/10/1287 (GA ČR).

References

[1] Unreal Technology: *Unreal Engine* [online]. [visited 2010-05-31]. URL: www.unrealtechnoogy.com.

[2] Valve Software: *Source Engine* [online]. [visited 2010-05-31]. URL: http://source.valvesoftware.com/

[3] M., O., Riedl, A., Stern: Believable Agents and Intelligent Scenario Direction for Social and Cultural Leadership Training. *Proceedings of the 15th Conference on Behavior Representation in Modeling and Simulation*, Baltimore. Maryland. 2006.

[4] J. Gemrot, C. Brom, R. Kadlec, M. Bída, O. Burkert, M. Zemcak, R. Pibil. Pogamut 3: Virtual Characters Made Simple. In J. Gray, and S. Nefti-Meziani, editors, Advances in Cognitive Systems. 2009

[5] Unreal Technology: *Unreal Tournament 3* [online]. [visited 2010-05-31]. URL: http://www.unrealtournament3.com.

[6] Virtual Battlespace 2 [online]. [visited 2010-05-31]. Bohemia Interactive. URL: http://virtualbattlespace.vbs2.com/

[7] IEEE Computer Society: 1516.4-2007 IEEE Recommended Practice for Verification, Validation, and Accreditation of a Federation - An Overlay to the High Level Architecture Federation Development and Execution Process. 2007.

[8] F. Kuhl, R. Weatherly, J. Dahmann: Creating Computer Simulation Systems: An Introduction to the High Level Architecture. Prentice Hall. 1999.