Dynamic Discovery of Simulation Entities Using Bamboo and HLA

CPT Stewart Liles
Kent Watsen
Michael Zyda
NPSNET Research Group
Naval Postgraduate School
Monterey, CA 93943

Keywords:
Bamboo, HLA, RTI

ABSTRACT: This paper describes a program module that administrates the RTI communication functions for the Bamboo virtual environment tool kit. The program module and its associated object model allow users to write Bamboo modules that dynamically load at runtime and promulgate to all members of the federation without explicit user interaction.

Bamboo is a portable architecture supporting arbitrary virtual environments. Its design focuses on the ability to dynamically configure itself without explicit user interaction, allowing applications to take on new functionality after execution. In particular this framework facilitates the discovery of virtual environments on the network at runtime.

Bamboo is ideally suited for its role in this HLA application. This simple demonstration virtual environment uses Bamboo and the developed HLA module to demonstrate the flexibility of Bamboo and HLA. The system uses a FOM that defines a simple object that is similar to a DIS entity state PDU. After the module loads and the simulation entity updates, the RTI object discovery mechanism updates all HLA administration modules in the federation. This unique use of the RTI allows large simulations to run in an ad hoc fashion allowing exercise coordinators greater flexibility in runtime configuration. Should a federate load a module that other members in the federation do not have local access to, Bamboo and the HLA administration module provide the means to retrieve and load the module from a remote server using the hypertext transfer protocol.

1. Introduction

The design and execution of a networked virtual environment (NVE) are challenging tasks made even more difficult by the fact that NVEs are becoming more complex and difficult to manage. In an HLA environment this includes the Federation Development Process (FEDEP). In a distributed environment a federate not only computes its own behaviors and publishes them to the network, but it also accurately represents all other federation entities participating in the simulation. HLA provides the network communication capabilities. Until now there was no way to ensure all federates had the proper polygonal and behavioral representation for all entities participating in the virtual environment. Bamboo provides such a capability by providing federates a framework to dynamically load and unload program modules as the situation changes in the virtual environment. By implementing federations as a group of program modules, designers solve the problem of ensuring that every site running in the federation is consistent with the every other. The designer just ensures every participant in the federation knows the network location of all the program modules making up the federation. Then, as the federation executes, each site loads and unloads modules as needed. All federates have the same representations for each entity as well as its associated behaviors and controls.

This paper describes an implementation that uses a new system called Bamboo to handle the dynamic nature of modern NVEs and HLA that handles the communication between federates. HLA is not discussed in detail; however, the following section provides an overview of Bamboo's features and how they apply to this implementation.

2. Bamboo

Bamboo enables dynamically scaleable virtual environments hosted on a network. It achieves this goal by an efficient implementation that provides direct support for the key issues pertaining to VE development. These issues include dynamic extensibility, multithreading and event handling. [1]

2.1 Dynamic Extensibility

Bamboo's most notable feature is its ability to dynamically extend its capabilities during run time. It achieves this goal by implementing the plug-in metaphor used by commercial packages like Netscape [2] and PhotoShop [3]. Bamboo then extends this metaphor by adding inter-module dependencies. Tracking inter-module dependencies could be a complex task. Fortunately, as Bamboo loads each module, it verifies that modules it depends on load first. If they are not, it automatically loads them without specific interaction with the user. Using Figure 1 as an example, assume M3 is already loaded. If M4 loads later, the system verifies the presence of M2 in memory. Bamboo loads M2 if it is not in memory. As M2 is being loaded Bamboo verifies the presence of M1. M4 finally loads because Bamboo verified all its dependencies [4].

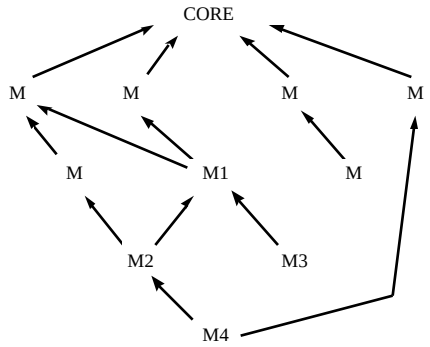

Figure 1: Module Dependency View

Dynamic loading of program modules does not in itself ensure dynamic extensibility. Each module has the opportunity to attach itself and remove itself from the process's execution loop. Bamboo uses a callback handler that allows each module to attach and remove itself from the process's execution loop when being paged in and out of memory. The callback handler derives from objects that can be named so it is easily located and manipulated. The callback itself is recursive and provides two callback handlers, one just before callback execution and one directly after. This allows grouping of like functionality. For example, rendering engines implement some form of app, cull and draw as a pipeline. Users refer to these areas as pre and post app, pre and post cull, and pre and post draw. The executable begins to resemble a tree of callbacks (see Figure 2). It follows, that any pruning or pausing of subtrees would automatically do the same to its children.

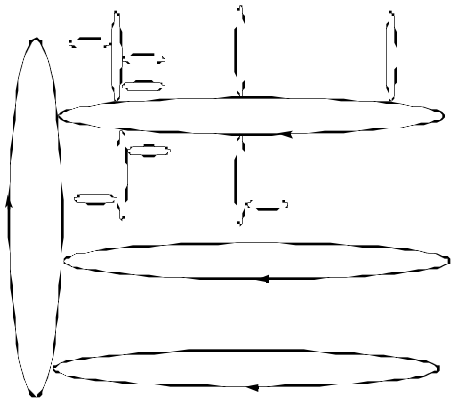

Figure 2: Extending the Executable

2.2 Multi-Threading

Bamboo also implements a system of light threads that cycle at a user defined rate. Each of these threads embeds a callback handler that is cycled when the thread is swapped in by the process scheduler. This mechanism simplifies the development of asynchronous operations while maintaining a consistent user interface.

2.3 Event Handling

The event handler simply provides an abstraction for handling system generated events. The event handler uses the callback handler to notify registered parties of an event. Bamboo receives this notification as a callback. Bamboo uses callback handlers so multiple callbacks respond to a single event.

3. Implementation

The implementation requires two major functions. There must be a system for communicating changes in player state from one work station to the next and for tracking all the players in the environment. The network communication system is the High Level Architecture (HLA) and the Run Time Infrastructure (RTI). They will not be discussed in detail except to describe how different services are used to accomplish the required communication tasks. The second system loads modules, captures user input, and passes the correct entity state information to the communication system. This mechanism essentially administrates the virtual environment at the work station level and ensures all functionality needed to execute the simulation is available to the user. That system is the HLA Administration Module (HLAAdmin).

Figure 3 shows the module dependency tree for this implementation. Notice that the HLAAdmin module depends on the Page module. This module allows users to load and unload modules during simulation execution. This figure also illustrates that all modules that represent simulation entities must depend on the HLAAdmin module in order to function as a member of the federation. This requirement means modules not written for the system will not be promulgated to the other federates.

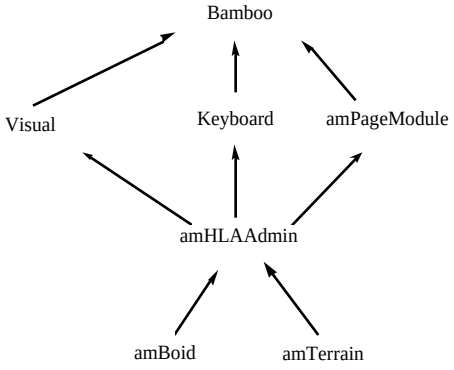

Figure 3: Implementation Dependency Model

3.1 HLA Administration Module

The HLAAdmin module is the main module and must be present on all workstations at the start of the simulation. The HLAAdmin module manages the following tasks: HLA federation management, loads and unloads system modules, opens the execution window, manages lists of the objects in the environment, and provides the mechanism for objects to update their state. Recall that Bamboo implements a system that enforces module dependencies, so all the above tasks are not part of the HLAAdmin module but are separate modules loaded at runtime or at the users request.

Federation management is this module's simplest task. Here the module creates and joins the federation. Next it publishes and subscribes to the objects and interactions needed for execution. Finally it provides a mechanism to register new simulation entities with the RTI.

The HLAAdmin module loads and unloads modules in two ways: either automatically at the request of the system or explicitly at the request of the user. The HLAAdmin module loads user requested modules using a separate module called the Page module on which it depends. This module loads automatically when the HLAAdmin module loads. The Page module's only task is to make the Bamboo calls that load and unload user requested modules. It also installs two event handlers tied to keyboard events that the user can make to accomplish the loads and unloads.

The HLAAdmin module manages of all the simulation entities. Functionality related to this task includes HLA object management tasks like registering and deleting objects and ensuring state updates transmit to the correct entity. We used a pure virtual base class that all objects inherent from to allow the HLAAdmin module to iterate its list of simulation entities and update the objects based on an identification number provided by the RTI.

Each module's capability means nothing without a model showing how Bamboo extends the executable in this implementation. Figure 4 illustrates the three execution threads used in this implementation. HLAAdmin module created the symbols in bold outline when the module loaded. A1 is a callback attached to the main callback handler created by the Bamboo core. A1 ticks the RTI providing CPU time to the RTI ambassador and the Federate ambassador. This callback drives the federate by processing all updates and providing them to the correct simulation entity. A2 is a callback attached to the draw callback handler of the Visual module. This callback calls the display function of all simulation objects using a call to a pure virtual function defined in the base object all simulation entities must implement. Finally AK is the callback representing all keyboard events that are processed by the HLAAdmin module.

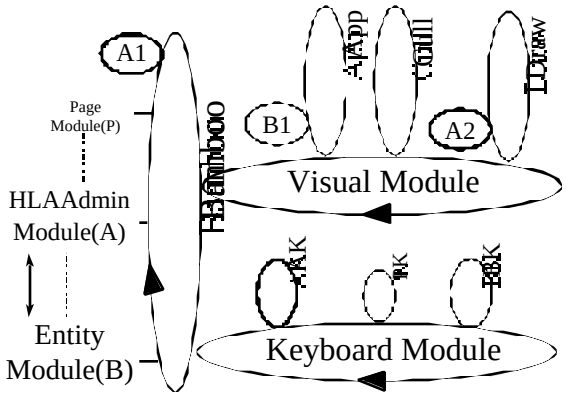

Figure 4: Execution Callback Tree

The HLAAdmin module is only a manager that administrates the federates participation in the simulation.

The simulation entities are the players and make up what the user interacts with during simulation execution.

3.2 Simulation Entities

As the VE executes, if an entity is updated that is not currently represented on the local machine, the RTI initiates the discovery process. The UserSuppliedTag, a character string that is transmitted with each update handle value pair, represents the module name. If this module is already loaded then another object from this module is instantiated. If the module does not exist, then Bamboo loads it and instantiates an object that represents the newly discovered entity.

Each simulation entity is a Bamboo module. Each module has two major components: the object's polygonal representation and its general behavior. Therefore, when a module loads as a result of a remote object update, the user collects all the controls of that module. Then Bamboo plugs the module into the local event loop so local processing can compute entity appearance and behavior. Figure 4 shows the entity module loaded and inserted in the executable with two sets of callbacks. B1 is the preapp callback that gives the user the ability to control the object with keyboard input. BK is the callback for all the keyboard inputs defined by the module.

Because this system passes behaviors along with polygonal representation we reduce network traffic by reducing the details of entity behavior that previous systems transmitted via the network. This occurs because each entity computes its behavior locally not from a remote location. For instance, each entity provides collision event behavior locally without the need for multiple interactions transmitted across the network. Now the entity module notifies the federation that a collision occurred not the detailed state changes resulting from the collision. Each entity computes those state changes locally as a result of the interaction. The result is a series of simulation actors whose behaviors and polygonal representations load dynamically at runtime. This allows simulation managers to easily experiment with the content of the environment by adding and subtracting functionality at runtime. The tendency is to think that this applies only to the graphically represented entities but it could mean that data loggers or analysis modules dynamically load and unload to collect and analyze simulation data. Bamboo provides an unprecedented method of adding functionality to an executing networked virtual environment.

3.3 Graphics Rendering

Bamboo's Visual module renders the graphical objects in the scene. The HLAAdmin module and the entity modules update the object's position and orientation. Each entity module registers callbacks with the Visual module to ensure accurate rendering of the simulation entity. These callbacks call the appropriate functions when the system needs to render the graphical representation of each entity. See Figure 4 for the callback tree representing this implementation.

4. Conclusions

The result of this work is a dynamically extensible distributed virtual environment that ensures consistency between distributed locations. Each federate operating in the simulation has an accurate representation of the environment without explicit interaction of the user. Users can add and delete functionality on an individual or federation wide level depending on the situation.

Bamboo is a highly extendible tool that finally allows designers the flexibility to design networked virtual environments without an overarching, monolithic structure that is unchangeable after it is compiled. Bamboo provides the flexibility to design large scale distributed environments in a modular fashion. This gives the user the ability to decide during simulation execution how to represent the environment and its actors.

5. Acknowledgments

This work is the result of many extended conversations with the co-authors and colleagues at the Naval Postgraduate School. It would never have compiled without the help of Kent Watsen, who also created Bamboo. Furthermore, the patience of Dr. Mike Zyda is appreciated. Finally, this effort is made possible by our sponsors: DMSO, DARPA and Advanced Network and Services.

6. Web Pointers

NPSNET Research Group
http://www.npsnet.nps.navy.mil

Bamboo
http://www.npsnet.nps.navy.mil/Bamboo

Implementation
http://www.stl.nps.navy.mil/~swliles

7. References

[1] Watsen, K. and M. Zyda (1998), Bamboo - A Portable system for Dynamically Extensible, Real-time, Networked Virtual Environments. 1998 IEEE Virtual Reality International Symposium (VRAIS '98), Atlanta, Georgia.

[2] Adobe(1997). Photoshop Software Development Kit, ftp://ftp.adobe.com/pub/adobe/devrelations/sdk/photoshop.

[3] Netscape(1997). Navigator 4.0 Plug-inGuide, http://developer.netscape.com/library/documentation/communicator/plugin/contents.html

[4] Watsen, K. and M. Zyda (1998), Bamboo - Supporting Dynamic Protocols for Virtual Environments. 1998 IMAGE Conference, Scottsdale, Arizona.

[5] High Level Architecture Run-Time Infrastructure Programmer's Guide Version 1.0, 15 May 1997

Author Biographies

CPT STEWART W. LILES is pursuing a M.S. degree in Modeling Virtual Environments and Simulation under Professor Zyda. CPT Liles received his B.S. from Oklahoma State University in 1988 in Management Sciences and Computer Systems. After graduation he has spent the next eight years in the Army as an Ordnance Officer until arriving at the Naval Postgraduate School in 1996. CPT Liles' last assignment was in command of the 178th Maintenance Company, Ft. Lewis Washington, responsible for the maintenance and spare parts supply of a PATRIOT Air Defense Battalion. He can be emailed at liless@cs.nps.navy.mil.

KENT WATSEN is pursuing a Ph.D. under Professor Mike Zyda while acting as project manager of the NPSNET Research Group in the Computer Science department at the Naval Postgraduate School in Monterey. He is the lead architect and developer of Bamboo, a virtual environment toolkit supporting, among other things, the next generation of NPSNET. His relevant experience includes the design and development of the character animation and 3D ocean modules for EasyScene, another virtual environment toolkit that he co-developed while working with Coryphaeus Software. He also developed Visual World, the rendering engine for a DIS simulator, while with DCS Corporation. Finally, He is responsible for a raytracing-for-animation package developed as his undergraduate thesis. He holds Computer Science and Applied Mathematics engineering degrees from the University of Virginia. He is currently a co-chair of the VRML symposium and is actively publishing and presenting papers at both IEEE and ACM sponsored conferences. He can be emailed at kent@watsen.net.

MICHAEL ZYDA is a Professor in Department of Computer at the Naval Postgraduate School, Monterey, California. Professor Zyda is also the Academic Associate and Chair of the NPS Modeling, Virtual Environments and Simulation curriculum. He has been at NPS since February of 1984. Professor Zyda's main focus in research is in the area of computer graphics, specifically the development of large-scale, networked 3D virtual environments. Professor Zyda was a member of the National Research Council's Committee on Virtual Reality Research and Development. Professor Zyda was the chair of the National Research Council's Computer Science and Telecommunications Board Committee on Modeling and Simulation: Linking Entertainment & Defense. Professor Zyda is also the Senior Editor for Virtual Environments for the MIT Press quarterly PRESENCE, the journal of teleoperation and virtual environments. He is a member of the Editorial Advisory Board of the journal Computers & Graphics. Professor Zyda is also a member of the Technical Advisory Board of the Fraunhofer Center for Research in Computer Graphics, Providence, Rhode Island. Professor Zyda has been active with the Symposium on Interactive 3D Graphics and was the chair of the 1990 conference, held at Snowbird, Utah and the chair of the 1995 Symposium, held in Monterey, California. Professor Zyda began his career in Computer Graphics in 1973 as part of an undergraduate research group, the Senses Bureau, at the University of California, San Diego. Professor Zyda received a BA in Bioengineering from the University of California, San Diego in La Jolla in 1976, an MS in Computer Science/Neurocybernetics from the University of Massachusetts, Amherst in 1978 and a DSc in Computer Science from Washington University, St. Louis, Missouri in 1984. He can be emailed at zyda@siggraph.org.