Building Multiple, Shared Environments with Entity Projection and Transportation Facilities

The MOSES Project:

Meta Operating System And Entity Shell

Daniel J. Pezely

18 April 1991

---

1 Introduction

An environment in the context of the MOSES Project means an application using the meta operating system and its libraries. Such applications may initiate any task and have the resources of the DataSpace and communication facilities available to them. Of course, security issues arise in the minds of many who read that last sentence, but such matters are easily dealt with and are explained in one of the following sections. However, our purpose here is to demonstrate to client programmers one possible way to build an environment. The environment described in this paper will also demonstrate how the distributive elements within MOSES work and describe the communication protocol suite use.

2 The Environment Desired

Think of a room with lots of doors. This room will be the space, or environment, which we intend create, and it may be as simple or as complex as we wish simply by adding information to the room description.

The doors in this room will each lead to a different environment, the nature of which are irrelevant. However, here, the doors are something from the *Twilight Zone* since upon traversing through any one of the doorways, the traveler is not on the other side of a physical door but is in another environment which could be physically thousands of miles away, thanks to the magic of wide-area internetworking.

The doorways may be seen as network connections, and the different environments may be seen as remote hosts on the network, since that is what they indeed are.

Let us add windows to this space so we can see into the different environments which we may want to enter. Now these windows give us a way to observe without traveling but we cannot interact with what we are observing. But can we interact without traveling also?

Being able to participate without leaving our space is something which many of us do all the time, but some don't realize it. This projected participation is done via remotely logging on to multiple hosts at the same time. The user is not physically multiplying him/herself to use the various hosts of course, but projecting his or her user identity across various connections to those hosts.

In the room analogy, projection can be thought of as telepresence: the use of robotics controlled remotely. Who- or what- ever is in the room can easily make a virtual-body clone as the robot and transport that to one of the different environments.

However, these virtual-body clones have a wonderful side-effect when used. The clones can be altered in any way when used, but as expected, the entity controlling the clone is not altered. Also, any alterations which may be made are local to that environment unless a *negotiation* is made with the controlling entity.

By negotiation, we mean that if the two parties involved do not communicate the changes or synchronize activities, the changes or activities do not actually happen. This is a complex topic which is explained in depth in a subsection below. See [1] for more philosophical discussions.

We have the environment, but we must also be able to move an entity around and be able to select which door or window we wish to traverse or look through. And of course, we must be able to create those clones of the entity so we can project into the different environments, possibly some or all at the same time.

Now, we have raised the following questions.

1. How is such an environment created within the domain of MOSES?
2. How do we transport the entity into one of the different environments?
3. How do we control the projections, and how can we deal with multiple projections?

These questions will be answered for those unfamiliar with virtual environments in the Design Overview section, and the more technical details are given in the Implementation section.

3 Design Overview

The heart of any environment to be created is a program. However, this program can be an executable which runs concurrently with MOSES or it can be a program which MOSES runs. One phrase which applies over and over again, is *it can, but it does not have to.* That is, the environment can be running concurrently, but it does not have to.

MOSES provides the client programmer with the function library to handle the mechanisms and access methods of storage, initiating a new task, and communication. These three objects are the fundamental elements of an *entity.*

As described in [2], everything is an entity: the environment, the program which defines the environment, the being within the environment, the components of the virtual-body, the different environments to choose from, and even the communications protocol data are entities.

3.1 Entities as Devices

Device entities [1] will be required to specify the movement and to provide a means of selection. These devices may be driven by actual, physical hardware or may be virtual, algorithmic devices. Devices (e.g., entities) may be empowered by a number of means. The important part of a device entity to remember is that it can be hardware-driven, but it does not have to.

The doorways to the environments, being entities too, can all lead to the different environments, different places within one environment, or all to the same place within one environment but perhaps with different attributes. The connected environment could even be the same environment which the entity would be leaving from.

The point to specifying some of the environment possibilities is that they are virtual; thus, they can be anything, or they do not have to be anything at all.

The following two subsections are obvious to most readers, but for the few who need the information, here it is in the context of MOSES.

3.1.1 Actual Devices

Actual devices refer to those elements of the system which there is hardware attached. In most cases, a software device driver will be needed or a high-level of access is provided (such as via file-like operations: open, close, read, write, etc.).

Sometimes, standard device drivers are too primitive, and in such cases, virtual devices are created which extend the functionality of the hardware device.

[1] Entities are entities, and there are no classifications of different types of entities; however, for clarity in this overview, we shall specify the nature of what the entity shall be used for.

3.1.2 Virtual Devices

3.2 The Virtual-Body

3.3 Communications Traffic

3.4 Negotiation

3.5 Security

4 Implementation Strategy

When time permits, this will be filled in.
No problems need to be overcome, other than finding the time...

5 Communications Protocol

References

[1] Bob Jacobson and John Barlow and Esther Dyson and Timothy Leary and William Bricken and Warren Robinett and Jaron Lanier. "Hip, hype and hope – The three faces of virtual worlds," *Computer Graphics (SIGGRAPH '90 Panel Proceedings)*, vol. 24, pp. 10.1-10.29 (August 1990). Edited by Alyce Kaprow.

[2] Pezely, D.J., *The MOSES Project: Implementation Design*, Human Interface Technology Laboratory, Washington Technology Center, University of Washington, Seattle, WA, 1991.