Design

Figure 1 (go ahead, look at it now) illustrates the preliminary **Distributed Game System Architecture**. This is the underlying support framework for the **Universe**. The**Uni verse** is distributed between two systems: the *Player's System* on the one hand and the *LFL System* on the other. Actually, there is only one LFL System, which we'll refer to as *Universe Central*, but there may be any number of independent Players' Systems (see figure 2).

Player's System

The Player's System consists of two major components, called the *foundation system* and the *viewpoint module*. The viewpoint module presents to the player a view into the **Universe**, corresponding to some role that the player may take or situation that the player may find himself in. The foundation system acts as a backend for the frontend presented by the viewpoint module, mediating information query and update requests between the local system and Universe Central. In addition, the viewpoint module encapsulates variations in display and input hardware, while the foundation system encapsulates variations in communications and data storage hardware. These two components are separate products, because there is potentially a large number of different possible viewpoint modules corresponding to the large number of possible viewpoints.

The viewpoint module consists of two sub-units: the *situation simulator* and the *display handler*. The display handler contains hardware-specific software for dealing with particular display devices, sound generators, and input devices. There will be a separate display handler tailored to each machine that we wish to support as a player system. The situation simulator contains interactive software that puts the player into a particular *situation* in the **Universe**. The range of possible situations is infinite, but obvious possibilities include such things as ''Merchant Spaceship Captain'', ''Planetary Emperor'', ''Space Fighter Pilot'', ''Pirate Captain'', ''Shipping Magnate'', ''Interstellar Banker'', ''Starship Designer'', ''Diplomatic Courier'', and so on. Each situation simulator would be marketed as a separate product, allowing for a wide variety of separate situations. This has obvious possibilities for marketing strategies to promote an active and interested long-term aftermarket in add-on modules, possibly some produced by third-party vendors.

The foundation system has three sub-units: the *database server*, the *resident database*, and the *communications channel controller*. Collectively these pieces form an application independent database backend for use by the viewpoint module. The database server is the most central of the three. It provides an interface between the viewpoint module and the **Universe** distributed database. It stores and retrieves information according to requests given to it by the viewpoint module. It speaks sometimes to the resident database, over which it has complete control, and sometimes to Universe Central via the communications channel controller. The resident database is simply a collection of information stored locally in the player's computer. The database server manages the caching of information in this local database in an effort to minimize the number of transactions with Universe Central, the assumption being that transactions with the local database are quick and inexpensive while transactions with Universe Central are slower and more costly. The communications channel controller is the gateway to Universe Central. It handles the arrangements for establishing communications over various possible channels (phone lines, cable TV, etc.) using a specialized communications protocol. It also performs on-the-fly data compression and expansion procedures to reduce communications bandwidth requirements.

LFL System

The LFL System, Universe Central, has a slightly more complicated structure as befits its centralized role in the **Universe**. At its heart is the*non-r esident database* (which is to say that it is not stored inside the players' machines and is therefor *non-resident from the players' point of view*). This is a body of information representing the common fabric of the **Universe** that all players share. Tapping into the non-resident database are two sorts of *access servers* which allow various sorts of manipulations of the database's contents. These are the*player access server* and the *GM access server*.

It is through the player access server that the players interact with the central database. The players' systems and the player access server do not interact with each other directly. Rather, their communications are mediated by communications channel controllers at both ends of the communications lines joining

them. Thus there are communications channel controllers at Universe Central which are duals of the ones in the players' systems.

The GM access server provides access to the non-resident database for the system administrators and gamemasters (*GMs* — roles to be described shortly). Layered on top of the GM access server are one or more *design tools* that enable the administrators and GMs to construct worlds, create game scenarios, administer player accounts, and so on.

The main distinction between the two access servers is in the nature of the manipulations that they support. The player access server only allows operations which are consistent with the established physical laws and existing structure of the **Universe**, whereas the GM access server allows any sort of wholesale alterations to the database that its users desire. This dichotomy is primarily a security measure, so that runaway bugs or malfunctions in players' machines (or malicious hackers attacking the system, for that matter) cannot ''rend the fabric of space-time'', so to speak.

The Interstellar Metaphor for Distributed Systems

Various contemporary office systems, such as Apple's Macintosh and Lisa, Xerox's Star workstation, and their imitators, make use of what has come to be called the ''desktop metaphor''. The idea is that these systems try to couch the complicated controls of a complicated computer system in terms of the (presumably) more familiar imagery and conceptual structures of the office workplace. These systems have gained much attention because of their innovative user interfaces, and, perhaps more importantly, inspired many imitators (some capable and most not). However, the excitement and controversy, publicized by a trade press which, by and large, has no idea what is really going on, has focused on the desktop metaphor itself, rather than on the deeper principles and findings which lead to its development.

To my mind, there are two ideas that are the essential findings of the Xerox human factors experts who originally gave us the desktop metaphor. First of all, they tell us, it is helpful, when trying to comprehend a complex system, to have a working model in one's head of what the system does and how it works. Most people find this a reasonable (if not obvious) assertion. The second idea, however, less intuitive. This idea is that the actual system design itself — the model that the designers and implementors have in *their* heads — is often *not* the best model for a user to have in order to learn and effectively use the system, even though it is, in some sense, the most ''accurate'' model.

The lesson to be drawn from this is that in the design of a large, complex system (which is what we are doing here), it can be constructive to have a metaphor or model in mind for the users to think about the system with. This can be both an aid in building an effective, easy to use user interface and as a frame of reference for assessing the consistency of the design and the behavior of the finished product.

What relevance, then, does this have to us? Consider the problems associated with building distributed systems. State synchronization among multiple networked processors is a major problem because communications, no matter how fast, are never instantaneous. Most individual processors can be in communication with only a small number of other processors (usually one) at a time, thus any given processor is unable to communicate with most of the network most of the time. It may, in fact, not even be aware of the presence of most of the network. Add to this the complication that processors may be appearing on or disappearing from the network at whim, that some may drop out of contact for lengthy periods due to hardware breakdowns or someone's failure to pay the phone bill, that the communications channels are often unreliable and may, in fact, fail in the midst of some transaction, and so on, and you have a real mess on your hands. Now, try explaining all these complications to the typical user.

The interstellar simulation I am proposing is, in fact, a fairly good, if somewhat ostentatious, model for distributed systems, provided that certain principles from physics and conventions from science fiction are adopted. Think of the isolated processors as worlds, separated by space. The speed of light and the effects of relativity complicate communications and transport. Of course, there is always hyperdrive to get from place to place, but sometimes weird things happen out in those extra dimensions... The point of this is that the limitations of the distributed medium can be written into the premise of **Universe** so that the players have a consistent model of what's going on without becoming confused. The software in the players' systems can be constructed to fudge the facts convincingly when things don't always go as planned. This metaphor also helps to blur the line between the fantasy (the exciting **Universe** with its spaceships to fly

and exotic worlds to explore) and the underlying reality (your home computer talking to my big computer over the phone) so that the medium becomes more transparent and less obtrusive.