internal change or on a periodic, timed basis. In this way, the game server can generate characters in the game or world that aren't controlled by some outside player.

This sort of programming model fits well with games and virtual worlds, but is also used in a number of enterprise-level architectures, such as J2EE and web services. The need to build an architecture different from those enterprise mechanisms was dictated by the very different environment in which MMOs and virtual worlds exist. This environment is nearly a mirror image of the classic enterprise environments, which means that if you have been trained in the enterprise environment, almost everything you know is going to be wrong in this new world.

The classic enterprise environment is envisioned as a thin client connected to a thick server (which is itself often connected to an even thicker database server). The server will hold most of the information needed by the clients, and will act as a filter to the backend database. Very little state is held at the client; in the best case, the client has very little memory, no disk of its own, and is a highly competent display device for the server, which is where most of the real work occurs.

The Game World

The MMO and virtual world environment starts with a very thick client—typically a top-of-the-line PC with the most powerful CPU available, lots of memory, and a graphics card that is itself computationally excellent, or a game console that is specially designed for graphics-intensive, highly interactive tasks. As much as possible, data is pushed out to these clients, especially data that is unchanging, such as geographic information, texture maps, and rule sets. The server is kept as simple as possible, generally holding a very abstract representation of the world and the entities within that world. Further, the server is designed to do as little computation as possible. Most of the computation goes on at the client. The real job of the server is to hold the shared truth of the state of the world, ensuring that any variation in the view of the world held at the various clients can be corrected as needed. The truth needs to be held by the server, since those who control the clients have a vested interest in maximizing their own performance, and thus might be tempted to change the shared truth (if they could) in their favor. Put more directly, players will cheat if they can, so the server must be the ultimate source of shared truth.

The data access patterns of MMOs and virtual worlds are also quite different from those that are seen in enterprise situations. The usual rule of thumb within the enterprise is that 90% of data accesses will be read-only, and most tasks read a large amount of data before altering a small amount. In the MMO and virtual world environment, most tasks access only a very small amount of the state on the server, but of the data that they access, about half of it will be altered.