MicroCosm Host Databases

What we store in the host
by
Chip Morningstar

Lucasfilm Ltd. Games Division
January 20, 1986

Introduction

This document is a rough requirements list for the **MicroCosm** host databases. It describes the various databases and what they must contain. It is intended to be advisory to Quantum in their preparation of the exact internal data representation.

The Databases

A very abstract representation of the world could express the entire contents of the **MicroCosm** in terms of generic objects and containment relationships between them. This would be the approach to follow in a system built on top of, say, Smalltalk-80. However, we are using a more prosaic technology and furthermore we need to be quite efficient. We will therefor recognize a few important special cases and represent the world by having the host maintain four basic databases:

- The Region Database
- The Avatar Database
- The Object Class Database
- The Object Database

Region Database

The Region Database contains one entry for each region in the **MicroCosm** (see the documents *MicroCosm Coordinate Systems and Topology* and *A Mapmaker's Guide to the MicroCosm*). Each region record contains:

- Region number (an integer, size currently indeterminate; primary key for region database)
- Name (a string; **optional**)
- Region X size (a byte)
- Region Y size (a byte)
- Terrain type (a byte)
- West neighbor (a pointer to another region or NIL)
- East neighbor (a pointer to another region or NIL)
- North neighbor (a pointer to another region or NIL)
- South neighbor (a pointer to another region or NIL)
- Owner (a pointer to an avatar or indicator that region is public)
- Contents (a list of pointers to objects)

Avatar Database

The Avatar Database contains one entry for each avatar currently living in the **MicroCosm**. Each Q-Link user name may have an avatar associated with it (or, more to the point, no user name may have more than one avatar). Each avatar record contains:

- Avatar number (an integer, size currently indeterminate; primary key for avatar database)
- Location, consisting of
  - ◻ Region (a pointer to a region)
  - ◻ X position (a byte)
  - ◻ Y position (a byte)
- Customization vector (several bytes, exact number currently undecided)
- Owner (a user name or user account number; another key for the avatar database)
- Contents (a list of pointers to objects)
- Name (a string; **optional**)

Object Class Database

The Object Class Database contains one entry for each type of object found in the **MicroCosm** (see the document *MicroCosm Minimal Object Set*). Each object is a member of some class of objects all of which have the same behavior protocol. This database thus contains all object information which is invariant across instances of a given class of objects. Each object class record contains:

- Class number (two bytes: a terrain code and a class code; primary key for object class database).
- Class name (a string; **optional**)
- Instance property record size (an integer)
- Important instance property record size (an integer)
- Capability vector (an array of procedure entry points)
- Capability routines (a series of executable procedures; probably not stored in the database itself but merely referenced by the capability vector)
- Imagery (a set of animation frames or static images, suitable for downloading ;basically an array of bytes)
- Imagery last changed date (a time stamp of some sort)
- Home system behavior (a collection of C64 code, suitable for downloading ;basically an array of bytes)
- Behavior last changed date (a time stamp of some sort)

Object Database

The Object Database contains one entry for each object extant in the **MicroCosm**. It contains the state-dependent properties of each object. There are a few general properties which all objects contain. However, the remaining properties depend on the class of the object. Each object record contains:

- Object identifier (an integer, size to be determined; primary key for object database)
- Short identifier (a byte or NIL)
- Class (a class number)
- Location, consisting of
  - ◻ Container (a pointer to a region or an avatar or an object)
  - ◻ X position (a byte)
  - ◻ Y position (a byte)
- Owner (a pointer to an avatar or indicator that object is public)
- Properties (a series of bytes whose length and interpretation depends upon the object's class)