

Journal of • Virtual Worlds Research

jvwresearch.org ISSN: 1941-8477

Vol.1. No.1

ISSN: 1941-8477

“Virtual Worlds Research: Past, Present & Future”

July 2008

The Lessons of Lucasfilm's Habitat¹

By Chip Morningstar and F. Randall Farmer; Independent Researchers

Abstract

Habitat is a "multi-player online virtual environment", created by Lucasfilm Games, a division of LucasArts Entertainment Company, in association with Quantum Computer Services, Inc. It was arguably one of the first attempts to create a very large scale commercial multi-user virtual environment in 1985. The system we developed could support a population of thousands of users in a single shared cyberspace. Habitat presented its users with a real-time animated view into an online simulated world in which users could communicate, play games, go on adventures, fall in love, get married, get divorced, start businesses, found religions, wage wars, protest against them, and experiment with self-government.

Our experiences developing the Habitat system, and managing the virtual world that resulted, offer a number of interesting and important lessons for prospective cyberspace architects. The purpose of this paper is to discuss some of these lessons. We hoped that the next generation of builders of virtual worlds can benefit from our experiences and (especially) from our mistakes.

Keywords: Habitat, Lucasfilm, virtual worlds, virtual environment, cyberspace.

This work is copyrighted under the Creative Commons Attribution-No Derivative Works 3.0 United States License by the Journal of Virtual Worlds Research.

¹ This paper was presented at The First International Conference on Cyberspace held in May 1990 at the University of Texas at Austin. It was published in *Cyberspace: First Steps*, Michael Benedikt (Ed.). (1991). Cambridge, MA: MIT Press.

The Lessons of Lucasfilm's Habitat²

By Chip Morningstar and F. Randall Farmer; Independent Researchers

Lucasfilm's Habitat was created by Lucasfilm Games, a division of LucasArts Entertainment Company, in association with Quantum Computer Services, Inc. It was arguably one of the first attempts to create a very large scale commercial multi-user virtual environment. A far cry from many laboratory research efforts based on sophisticated interface hardware and tens of thousands of dollars per user of dedicated compute power, Habitat is built on top of an ordinary commercial online service and uses an inexpensive—some would say "toy"—home computer to support user interaction. In spite of these somewhat plebeian underpinnings, Habitat is ambitious in its scope. The system we developed can support a population of thousands of users in a single shared cyberspace. Habitat presents its users with a real-time animated view into an online simulated world in which users can communicate, play games, go on adventures, fall in love, get married, get divorced, start businesses, found religions, wage wars, protest against them, and experiment with self-government.

The Habitat project proved to be a rich source of insights into the nitty-gritty reality of actually implementing a serious, commercially-viable cyberspace environment. Our experiences developing the Habitat system and managing the virtual world that resulted, offer a number of interesting and important lessons for prospective cyberspace architects. The purpose of this paper is to discuss some of these lessons. We hope that the next generation of builders of virtual worlds can benefit from our experiences and (especially) from our mistakes.

Due to space limitations, we will not be able to go into as much technical detail as we might like; this will have to be left to a future publication. Similarly, we will only be able to touch briefly upon some of the history of the project as a business venture, which is a fascinating subject of its own. Although we will conclude with a brief discussion of some of the future directions for this technology, a more detailed exposition on this topic will also have to wait for a future article.

The essential lesson that we have abstracted from our experiences with Habitat is that a cyberspace is defined more by the interactions among the actors within it than by the technology with which it is implemented. While we find much of the work presently being done on elaborate interface technologies—DataGloves, head-mounted displays, special-purpose rendering engines, and so on—both exciting and promising, the almost mystical euphoria that currently seems to surround all this hardware is, in our opinion, both excessive and somewhat misplaced. We cannot help having a nagging sense that it is all a bit of a distraction from the really pressing issues. At the core of *our* vision is the idea that cyberspace is necessarily a *multiple-participant environment*. It seems to us that the things that are important to the inhabitants of such an environment are the capabilities available to them, the characteristics of the other people they

² This paper was presented at The First International Conference on Cyberspace held in May 1990 at the University of Texas at Austin. It was published in *Cyberspace: First Steps*, Michael Benedikt (Ed.). (1991). Cambridge, MA: MIT Press.

encounter there, and the ways these various participants can affect one another. Beyond a foundation set of communications capabilities, the details of the technology used to present this environment to its participants, while sexy and interesting, are of relatively peripheral concern.

What is Habitat?

Habitat is a "multi-player online virtual environment" (its purpose is to be an entertainment medium; consequently, the users are called "players"). Each player uses his or her home computer as a frontend, communicating over a commercial packet-switching data network to a centralized backend system. The frontend provides the user interface, generating a real-time animated display of what is going on and translating input from the player into requests to the backend. The backend maintains the world model, enforcing the rules and keeping each player's frontend informed about the constantly changing state of the universe. The backend enables the players to interact not only with the world but with each other.

Habitat was inspired by a long tradition of "computer hacker science fiction", notably Vernor Vinge's novel, *True Names* (Vinge, 1981), as well as many fond childhood memories of games of make-believe, more recent memories of role-playing games and the like, and numerous other influences too thoroughly blended to pinpoint. To this we added a dash of silliness, a touch of cyberpunk (Gibson, 1984; Sterling, 1986), and a predilection for object-oriented programming (Sussman and Abelson, 1985).

The initial incarnation of Habitat uses a Commodore 64 for the frontend¹ Figure 1 is a typical screen from this version of the system. The largest part of the screen is devoted to the graphics display. This is an animated view of the player's current location in the Habitat world. The scene consists of various objects arrayed on the screen, such as the houses and tree you see here. The players are represented by animated figures that we call "Avatars". Avatars are usually, though not exclusively, humanoid in appearance. In this scene you can see two of them, carrying on a conversation.

Avatars can move around, pick up, put down and manipulate objects, talk to each other, and gesture—each under the control of an individual player. Control is through the joystick, which enables the player to point at things and issue commands. Talking is accomplished by typing on the keyboard. The text that a player types is displayed over his or her Avatar's head in a cartoon-style "word balloon."



Figure 1 -- A typical Habitat scene
(© 1986 LucasArts Entertainment Company).

The Habitat world is made up of a large number of discrete locations that we call "regions". In its prime, the prototype Habitat world consisted of around 20,000 of them. Each region can adjoin up to four other regions, which can be reached simply by walking your Avatar to one or another edge of the screen. Doorways and other passages can connect to additional regions. Each region contains a set of objects that define the things that an Avatar can do there and the scene that the player sees on the computer screen.

Some of the objects are structural, such as the ground or the sky. Many are just scenic, such as the tree or the mailbox. Most objects, however, have some function that they perform. For example, doors transport Avatars from one region to another and may be opened, closed, locked, and unlocked. ATMs (Automatic Token Machines) enable access to an Avatar's bank account². Vending machines dispense useful goods in exchange for Habitat money. Many objects are portable and may be carried around in an Avatar's hands or pockets. These include various kinds of containers, money, weapons, tools, and exotic magical implements. Table 1 lists some of the most important types of objects and their functions. The complete list of object types numbers in the hundreds.

Many objects are portable and may be carried around in an Avatar's hands or pockets. These include various kinds of containers, money, weapons, tools, and exotic magical implements. Listed here are some of the most important types of objects and their functions. The complete list of object types numbers in the hundreds.

<i>Object Class</i>	<i>Function</i>
ATM	Automatic Token Machine; access to an Avatar's bank account
Avatar	Represents the player in the Habitat world
Bag, Box	Containers in which things may be carried
Book	Document for Avatars to read (e.g., the daily newspaper)
Bureaucrat-in-a-box	Communication with system operators
Change-o-matic	Device to change Avatar gender
Chest, Safe	Containers in which things can be stored
Club, Gun, Knife	Various weapons
Compass	Points direction to West Pole
Door	Passage from one region to another; can be locked
Drugs	Various types; changes Avatar body state, e.g., cure wounds
Elevator	Transportation from one floor of a tall building to another
Flashlight	Provides light in dark places
Fountain	Scenic highlight; provides communication to system designers
Game piece	Enables various board games: backgammon, checkers, chess, etc.
Garbage can	Disposes of unwanted objects
Glue	System building tool; attaches objects together
Ground, Sky	The underpinnings of the world
Head	An Avatar's head; comes in many styles; for customization
Key	Unlocks doors and other containers
Knick-knack	Generic inert object; for decorative purposes
Magic wand	Various types, can do almost anything
Paper	For writing notes, making maps, etc.; used in mail system
Pawn machine	Buys back previously purchased objects
Plant, Rock, Tree	Generic scenic objects
Region	The foundation of reality
Sensor	Various types, detects otherwise invisible conditions in the world
Sign	Allows attachment of text to other objects
Stun gun	Non-lethal weapon
Teleport booth	Means of quick long-distance transport; analogous to phone booth
Tokens	Habitat money
Vendroid	Vending machine; sells things

Table 1 -- Some important object classes

Implementation

The following, along with several programmer-years of tedious and expensive detail that we will not cover here, is how the system works:

At the heart of the Habitat implementation is an object-oriented model of the universe. The frontend consists of a system kernel and a collection of objects. The kernel handles memory management, display generation, disk I/O, telecommunications, and other "operating system" functions. The objects implement the semantics of the world itself. Each type of Habitat object has a definition consisting of a set of resources, including animation cels to drive the display, audio data, and executable code. An object's executable code implements a series of standard behaviors, each of which is invoked by a different player command or system event. The model is similar to that found in an object-oriented programming system such as Smalltalk (Goldberg and Robson, 1983), with its classes, methods, and messages. These resources consume significant amounts of scarce frontend memory, so we cannot keep them all in core at the same time. Fortunately, their definitions are invariant, so we simply swap them in from disk as we need them, discarding less recently used resources to make room.

When an object is instantiated, we allocate a block of memory to contain the object's state. The first several bytes of an object's state information take the same form in all objects, and include such things as the object's screen location and display attributes. This standard information is interpreted by the system kernel as it generates the display and manages the run-time environment. The remainder of the state information varies with the object type and is accessed only by the object's behavior code.

Object behaviors are invoked by the kernel in response to player input. Each object responds to a set of standard verbs that map directly onto the commands available to the player. Each behavior is simply a subroutine that executes the indicated action; to do this it may invoke the behaviors of other objects or send request messages to the backend. Besides the standard verb behaviors, objects may have additional behaviors which are invoked by messages that arrive asynchronously from the backend.

The backend also maintains an object-oriented representation of the world. As in the frontend, objects on the backend possess executable behaviors and in-memory state information. In addition, since the backend maintains a persistent global state for the entire Habitat world, the objects are also represented by database records that may be stored on disk when not "in use". Backend object behaviors are invoked by messages from the frontend. Each of these backend behaviors works in roughly the same way: a message is received from a player's frontend requesting some action; the action is taken and some state changes to the world result; the backend behavior sends a response message back to the frontend informing it of the results of its request and notification messages to the frontends of any other players who are in the same region, informing *them* of what has taken place.

The Lessons

In order to say as much as we can in the limited space available, we will describe what we think we learned via a series of principles or assertions surrounded by supporting reasoning and

illustrative anecdotes. A more formal and thorough exposition will have to come later in some other forum where we might have the space to present a more comprehensive and detailed model.

We mentioned our primary principle above:

- **A multi-user environment is central to the idea of cyberspace.**

It is our deep conviction that a definitive characteristic of a cyberspace system is that it represents a multi-user environment. This stems from the fact that what (in our opinion) people seek in such a system is richness, complexity, and depth. Nobody knows how to produce an automaton that even approaches the complexity of a real human being, let alone a society. Our approach, then, is not even to attempt this, but instead to use the computational medium to augment the communications channels between real people.

If what we are constructing is a multi-user environment, it naturally follows that some sort of communications capability must be fundamental to our system. However, we must take into account an observation that is the second of our principles:

- **Communications bandwidth is a scarce resource.**

This point was driven home to us by one of Habitat's nastier, externally-imposed design constraints, namely that it only provided a satisfactory experience to the player over a 300 baud serial telephone connection (one routed, moreover, through commercial packet-switching networks that impose an additional, uncontrollable latency of 100 to 5000 milliseconds on each packet transmitted).

Even in a more technically advanced network, however, bandwidth remains scarce in the sense that economists use the term: available carrying capacity is not unlimited. The law of supply and demand suggests that no matter how much capacity is available, you always want more. When communications technology advances to the point where we all have multi-gigabaud fiber optic connections into our homes, computational technology will have advanced to match. Our processors' expanding appetite for data will mean that the search for ever more sophisticated data compression techniques will *still* be a hot research area (though what we are compressing may at that point be high-resolution volumetric time-series or something even more esoteric). (Drexler, 1986)

Computer scientists tend to be reductionists who like to organize systems in terms of primitive elements that can be easily manipulated within the context of a simple formal model. Typically, you adopt a small variety of very simple primitives which are then used in large numbers. For a graphics-oriented cyberspace system, the temptation is to build upon bit-mapped images or polygons or some other *graphic* primitive. These sorts of representations, however, are invitations to disaster. They arise from an inappropriate fixation on display technology, rather than on the underlying purpose of the system.

However, the most significant part of what *we* wish to be communicating are human behaviors. These, fortunately, can be represented quite compactly, provided we adopt a relatively abstract, high-level description that deals with behavioral concepts directly. This leads to our third principle:

- **An object-oriented data representation is essential.**

Taken at its face value, this assertion is unlikely to be controversial, as object-oriented programming is currently the methodology of choice among the software engineering *cognoscenti*. However, what we mean here is not only that you should adopt an object-oriented approach, but that the basic objects from which you build the system should correspond, more or less, to the objects in the user's conceptual model of the virtual world, that is, people, places, and artifacts. You could, of course, use object-oriented programming techniques to build a system based on, say, polygons, but that would not help to cope with the fundamental problem.

The goal is to enable the communications between machines take place primarily at the behavioral level (what people and things are doing) rather than at the presentation level (how the scene is changing). The description of a place in the virtual world should be in terms of what is there rather than what it looks like. Interactions between objects should be described by functional models rather than by physical ones. The computation necessary to translate between these higher-level representations and the lower-level representations required for direct user interaction is an essentially local function. At the local processor, display-rendering techniques may be arbitrarily elaborate and physical models arbitrarily sophisticated. The data channel capacities required for such computations, however, need not and should not be squeezed into the limited bandwidth available between the local processor and remote ones. Attempting to do so just leads to disasters such as NAPLPS (ANSI, 1983; Alber, 1985) which couples dreadful performance with a display model firmly anchored in the technology of the 1970s.

Once we begin working at the conceptual rather than the presentation level, we are struck by the following observation:

- **The implementation platform is relatively unimportant.**

The presentation level and the conceptual level cannot (and should not) be *totally* isolated from each other. However, defining a virtual environment in terms of the configuration and behavior of objects, rather than their presentation, enables us to span a vast range of computational and display capabilities among the participants in a system. This range extends both upward and downward. As an extreme example, a typical scenic object, such as a tree, can be represented by a handful of parameter values. At the lowest conceivable end of things might be an ancient Altair 8800 with a 300 baud ASCII dumb terminal, where the interface is reduced to fragments of text and the user sees the humble string so familiar to the players of text adventure games, "There is a tree here." At the high end, you might have a powerful processor that generates the image of the tree by growing a fractal model and rendering it three dimensions at high resolution, the finest details ray-traced in real time, complete with branches waving in the breeze and the sound of wind in the leaves coming through your headphones in high-fidelity digital stereo. And these two users might be looking at the same tree in same the place in the same world and talking to each other as they do so. Both of these scenarios are implausible at the moment; the first because nobody would suffer with such a crude interface when better ones are so readily available, the second because the computational hardware does not yet exist. The point, however, is that this approach covers the ground between systems already obsolete and ones that are as yet gleams in their designers' eyes. Two consequences of this are significant. The first is that we can build effective cyberspace systems today. Habitat exists as ample proof of this principle. The second is that it is conceivable that with a modicum of cleverness and foresight you could start building a

system with today's technology that could evolve smoothly as tomorrow's technology develops. The availability of pathways for growth is important in the real world, especially if cyberspace is to become a significant communications medium (as we obviously think it should).

Given that we see cyberspace as fundamentally a communications medium rather than simply an user interface model, and given the style of object-oriented approach that we advocate, another point becomes clear:

- **Data communications standards are vital.**

However, our concerns about cyberspace data communications standards center less upon data transport protocols than upon the definition of the data being transported. The mechanisms required for reliably getting bits from point A to point B are not terribly interesting to us. This is not because these mechanisms are not essential (they obviously are) nor because they do not pose significant research and engineering challenges (they clearly do). It is because we are focused on the unique communications needs of an object-based cyberspace. We are concerned with the protocols for sending messages between objects; that is, for communicating behavior rather than presentation and for communicating object definitions from one system to another.

Communicating object definitions seems to us to be an especially important problem, and one that we really did not have an opportunity to address in Habitat. It *will* be necessary to address this problem if we are to have a dynamic system in the future. Once the size of the system's user base has grown modestly large, it becomes impractical to distribute a new release of the system software every time one wants to add a new class of object. However, we feel the ability to add new classes of objects over time is crucial if the system is to be able to evolve.

While we are on the subject of communications standards, we would like to make some remarks about the ISO Reference Model of Open System Interconnection (ISO, 1986). This 7-layer model has become a centerpiece of most discussions about data communications standards today. It is so firmly established in the data communications standards community that it is virtually impossible to find a serious contemporary publication on the subject that does not begin with some variation on Figure 2. Unfortunately, while the bottom 4 or 5 layers of this model provide a more or less sound framework for considering data transport issues, we feel that the model's Presentation and Application layers are not so helpful when considering cyberspace data communications.

<u>Level</u>	<u>Function</u>
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Link
1	Physical

Figure 2 -- The 7-layer ISO Reference Model of Open System Interconnection.

We have two main quarrels with the ISO model. First, it partitions the general data communications problem in a way that is a poor match for the needs of a cyberspace system. Second, and more importantly, we think that the model itself is an active source of confusion because it focuses the attention of system designers on the wrong set of issues and thus leads them to spend their time solving the wrong set of problems. We know because this happened to us. "Presentation" and "Application" are simply the wrong abstractions for the higher levels of a cyberspace communications protocol. A "Presentation" protocol presumes that at least some characteristics of the display are embedded in the protocol. The discussions above should give some indication why we think that such a presumption is both unnecessary and unwise. Certainly, an "Application" protocol presumes a degree of foreknowledge of the message environment that is incompatible with the sort of dynamically evolving object system we envision.

A better model would be to substitute a different pair of top layers (Figure 3): a Message layer, which defines the means by which objects can address one another and standard methods of encapsulating structured data and encoding low-level data types (e.g., numbers); and a Definition layer built on top of the Message layer, which defines a standard representation for object definitions so that object classes can migrate from machine to machine. One might argue that these are simply Presentation and Application with different labels. However, the differences are so easily reconciled. In particular, we think the ISO model has, however unintentionally, systematically deflected workers in the field from considering many of the issues that concern us.

<u>Level</u>	<u>Function</u>
6	Definition
5	Message
4	Transport
3	Network
2	Link
1	Physical

Figure 3 -- A possible alternative protocol model.

World Building

There were two sorts of implementation challenge that Habitat posed. The first was the problem of creating a working piece of technology—developing the animation engine, the object-oriented virtual memory, the message-passing pseudo operating system, and squeezing them all into the ludicrous Commodore 64 (the backend system also posed interesting technical problems, but its constraints were not as vicious). The second challenge was the creation and management of the Habitat world itself. It is the experiences from the latter exercise that we think will be most relevant to future cyberspace designers.

We were initially our own worst enemies in this undertaking, victims of a way of thinking to which we engineers are dangerously susceptible. This way of thinking is characterized by the conceit that all things may be planned in advance and then directly implemented according to the plan's detailed specification. For persons schooled in the design and construction of systems based on simple, well-defined and well-understood foundation principles, this is a natural attitude to have. Moreover, it is entirely appropriate when undertaking most engineering projects. It is a

frame of mind that is an essential part of a good engineer's conceptual tool kit. Alas, in keeping with Maslow's assertion that, "to the person who has only a hammer, all the world looks like a nail," it is a frame of mind that is easy to carry beyond its range of applicability. This happens when a system exceeds the threshold of complexity above which the human mind loses its ability to maintain a complete and coherent model.

One generally hears about systems crossing the complexity threshold when they become very large. For example, the Space Shuttle and the B-2 bomber are both systems above this threshold, necessitating extraordinarily involved, cumbersome, and time-consuming procedures to keep the design under control—procedures that are at once vastly expensive and only partially successful. To a degree, the complexity of a problem can be dissolved by "throwing money" at it: faster computers, more managers, more bureaucratic procedures, and so on. However, such capital-intensive management techniques are a luxury not available to most projects. Furthermore, although these "solutions" to the complexity problem are out of reach of most projects, alas the complexity threshold itself is not. Smaller systems can suffer from the same sorts of problems. It is possible to push much smaller and less elaborate systems over the complexity threshold simply by introducing chaotic elements that are outside the designers' sphere of control or understanding. The most significant such chaotic elements are autonomous computational agents (e.g., other computers). This is why, for example, debugging even very simple communications protocols often proves surprisingly difficult. Furthermore, a special circle of living Hell awaits the implementors of systems involving that most important category of autonomous computational agents of all, groups of interacting human beings. This leads directly to our next (and possibly most controversial) assertion:

- **Detailed central planning is impossible. Don't even try.**

The constructivist prejudice that leads engineers into the kinds of problems just mentioned has received more study from economists, philosophers, and sociologists (e.g., Popper 1962, 1972; Hayek 1973, 1978, 1989; Sowell 1987) than from researchers in the software engineering community. Game and simulation designers are experienced in creating closed virtual worlds for individuals and small groups. However, they have had no reason to learn to deal with large populations of simultaneous users. Each user or small group is unrelated to the others and the same world can be used over and over again. If you are playing an adventure game, the fact that thousands of other people elsewhere in the (real) world are playing the same game has no effect on your experience. It is reasonable for the creator of such a world to spend tens or even hundreds of hours crafting the environment for each hour that a user will spend interacting with it, since that user's hour of experience will be duplicated tens of thousands of times by tens of thousands of other individual users.

Builders of today's online services and communications networks are experienced in dealing with large user populations, but they do not, in general, create elaborate environments. Furthermore, in a system designed to deliver information or communications services, large numbers of users are simply a load problem rather than a complexity problem. All the users get the same information or services; the comments in the previous paragraph regarding duplication of experience apply here as well. It is not necessary to match the size and complexity of the information space to the size of the user population. While it may turn out that the quantity of information available on a service is largely a function of the size of the user population itself, this information can generally be organized into a systematic structure that can still be maintained

by a few people. The bulk of this information is produced by the users themselves, rather than the system designers. (This observation, in fact, is the first clue to the solution to our problem.)

Our original, contractual specification for Habitat called for us to create a world capable of supporting a population of 20,000 Avatars, with expansion plans for up to 50,000. By any reckoning this is a large undertaking and complexity problems would certainly be expected. However, in practice we exceeded the complexity threshold very early in development. By the time the population of our online community had reached around 50 we were in over our heads (and these 50 were "insiders" who were prepared to be tolerant of holes and rough edges).

Moreover, a virtual world such as Habitat needs to scale with its population. For 20,000 Avatars we needed 20,000 "houses", organized into towns and cities with associated traffic arteries and shopping and recreational areas. We needed wilderness areas between the towns so that everyone would not be jammed together into the same place. Most of all, we needed things for 20,000 people to do. They needed interesting places to visit -- and since they can't all be in the same place at the same time, they needed a *lot* of interesting places to visit -- and things to do in those places. Each of those houses, towns, roads, shops, forests, theaters, arenas, and other places is a distinct entity that someone needs to design and create. Attempting to play the role of omniscient central planners, we were swamped.

Automated tools may be created to aid the generation of areas that naturally possess a high degree of regularity and structure, such as apartment buildings and road networks. We created a number of such tools, whose spiritual descendents will no doubt be found in the standard bag of tricks of future cyberspace architects. However, the very properties which make some parts of the world amenable to such techniques also make those same parts of the world among the least important. It is really not a problem if every apartment building looks pretty much like every other. It is a big problem if every enchanted forest looks the same. Places whose value lies in their uniqueness, or at least in their differentiation from the places around them, need to be crafted by hand. This is an incredibly labor-intensive and time consuming process. Furthermore, even very imaginative people are limited in the range of variation that they can produce, especially if they are working in a virgin environment uninfluenced by the works and reactions of other designers.

Running The World

The world design problem might still be tractable, however, if all players had the same goals, interests, motivations, and types of behavior. Real people, however, are all different. For the designer of an ordinary game or simulation, human diversity is not a major problem, since he or she gets to establish the goals and motivations on the participants' behalf and to specify the activities available to them in order to channel events in the preferred direction. Habitat, however, was deliberately open-ended and pluralistic. The idea behind our world was precisely that it did not come with a fixed set of objectives for its inhabitants, but rather provided a broad palette of possible activities from which the players could choose, driven by their own internal inclinations. It was our intention to provide a variety of possible experiences, ranging from events with established rules and goals (a treasure hunt, for example) to activities propelled by the players' personal motivations (starting a business, running the newspaper) to completely free-form, purely existential activities (hanging out with friends and conversing). Most activities, however,

involved some degree of planning and setup on our part. We were to be like the cruise director on a ocean voyage, but it turned out we were still thinking like game designers.

The first goal-directed event planned for Habitat was a rather involved treasure hunt called the "D'nalsi Island Adventure". It took us hours to design, weeks to build (including a 100-region island), and days to coordinate the actors involved. It was designed much like the puzzles in an adventure game. We thought it would occupy our players for days. In fact, the puzzle was solved in about 8 hours by a person who had figured out the critical clue in the first 15 minutes. Many of the players hadn't even had a chance to get into the game. The result was that one person had had a wonderful experience, dozens of others were left bewildered, and a huge investment in design and setup time had been consumed in an eyeblink. We expected that there would be a wide range of "adventuring" skills in the Habitat audience. What was not so obvious until afterward was that this meant that most people did not have a very good time, if for no other reason than that they never really got to participate. It would clearly be foolish and impractical for us to do things like this on a regular basis.

Again and again we found that activities based on often unconscious assumptions about player behavior had completely unexpected outcomes (when they were not simply outright failures). It was clear that we were not in control. The more people we involved in something, the less in control we were. We could influence things, we could set up interesting situations, we could provide opportunities for things to happen, but we could not predict nor dictate the outcome. Social engineering is, at best, an inexact science, even in proto-cyberspaces. Or, as some wag once said, "in the most carefully constructed experiment under the most carefully controlled conditions, the organism will do whatever it damn well pleases."

Propelled by these experiences, we shifted into a style of operations in which we let the players themselves drive the direction of the design. This proved far more effective. Instead of trying to push the community in the direction we thought it should go, an exercise rather like herding mice, we tried to observe what people were doing and aid them in it. We became facilitators as much as designers and implementors. This often meant adding new features and new regions to the system at a frantic pace, but almost all of what we added was used and appreciated, since it was well matched to people's needs and desires. As the experts on how the system worked, we could often suggest new activities for people to try or ways of doing things that people might not have thought of. In this way we were able to have considerable influence on the system's development in spite of the fact that we did not really hold the steering wheel—more influence, in fact, than we had had when we were operating under the delusion that we controlled everything.

Indeed, the challenges posed by large systems in general are prompting some researchers to question the centralized, planning-dominated attitude that we have criticized here and to propose alternative approaches based on evolutionary and market principles. (Miller and Drexler, 1988a, 1988b; Drexler and Miller 1988) These principles appear applicable to complex systems of all types, not merely those involving interacting human beings.

The Great Debate

Among the objects we made available to Avatars in Habitat were guns and various other sorts of weapons. We included these because we felt that players should be able to materially

effect each other in ways that went beyond simply talking, ways that required real moral choices to be made by the participants. We recognized the age-old storyteller's dictum that conflict is the essence of drama. Death in Habitat was, of course, not like death in the real world! When an Avatar is killed, he or she is teleported back home, head in hands (literally), pockets empty, and any object in hand at the time dropped on the ground at the scene of the crime. Any possessions carried at the time are lost. It was more like a setback in a game of "Chutes and Ladders" than real mortality. Nevertheless, the death metaphor had a profound effect on people's perceptions. This potential for murder, assault, and other mayhem in Habitat was, to put it mildly, controversial. The controversy was further fueled by the potential for lesser crimes. For instance, one Avatar could steal something from another Avatar simply by snatching the object out its owner's hands and running off with it.

We had imposed very few rules on the world at the start. There was much debate among the players as to the form that Habitat society should take. At the core of much of the debate was an unresolved philosophical question: is an Avatar an extension of a human being (thus entitled to be treated as you would treat a real person), or a Pac-Man-like critter destined to die a thousand deaths, or something else entirely? Is Habitat murder a crime? Should all weapons be banned? Or is it all "just a game"? To make a point, one of the players took to randomly shooting people as they roamed around. The debate was sufficiently vigorous that we took a systematic poll of the players. The result was ambiguous: 50% said that Habitat murder was a crime and should not be a part of the world, while the other 50% said it was an important part of the fun.

We compromised by changing the system to allow thievery and gunplay only outside the city limits. The wilderness would be wild and dangerous while civilization would be orderly and safe. This did not resolve the debate, however. One of the outstanding proponents of the anti-violence point of view was motivated to open the first Habitat church, the Order of the Holy Walnut (in real life he was a Greek Orthodox priest). His canons forbid his disciples to carry weapons, steal, or participate in violence of any kind. His church became quite popular and he became a very highly respected member of the Habitat community.

Furthermore, while we had made direct theft impossible, one could still engage in indirect theft by stealing things set on the ground momentarily or otherwise left unattended. And the violence still possible in the outlands continued to bother some players. Many people thought that such crimes ought to be prevented or at least punished somehow, but they had no idea how to do so. They were used to a world in which law and justice were always things provided by somebody else. Somebody eventually made the suggestion that there ought to be a Sheriff. We quickly figured out how to create a voting mechanism and rounded up some volunteers to hold an election. A public debate in the town meeting hall was heavily attended, with the three Avatars who had chosen to run making statements and fielding questions. The election was held, and the town of Populopolis acquired a Sheriff.

For weeks the Sheriff was nothing but a figurehead, though he was a respected figure and commanded a certain amount of moral authority. We were stumped about what powers to give him. Should he have the right to shoot anyone anywhere? Give him a more powerful gun? A magic wand to zap people off to jail? What about courts? Laws? Lawyers? Again we surveyed the players, eventually settling on a set of questions that could be answered via a referendum. Unfortunately, we were unable to act on the results before the pilot operations ended and the version of the system in which these events took place was shut down. It was clear, however, that

there are two basic camps: anarchists and statist. This division of characters and world views is an issue that will need to be addressed by future cyberspace architects. However, our view remains that a virtual world need not be set up with a "default" government, but can instead evolve one as needed.

A Warning

Given the above exhortation that control should be released to the users, we need to inject a note of caution and present our next assertion:

- **You can't trust *anyone*.**

This may seem like a contradiction of much of the preceding, but it really is not. Designers and operators of a cyberspace system must inhabit two levels of "virtuality" at once. The first we call the "infrastructure level," the level of implementation where the laws that govern "reality" have their genesis. The second we call the "experiential level," which is what the users see and interact with. It is important that there not be "leakage" between these two levels. The first level defines the physics of the world. If its integrity is breached, the consequences can range from aesthetic unpleasantness (the audience catches a glimpse of the scaffolding behind the false front), to psychological disruption (somebody does something "impossible," thereby violating users' expectations and damaging their fantasy), to catastrophic failure (somebody crashes the system). When we exhort cyberspace system designers to give control to the users, we mean control at the experiential level. When we say that you can't trust anyone, we mean that you can't trust them with access to the infrastructure level. Some stories from Habitat will illustrate this.

When designing a piece of software, you generally assume that the software is the sole intermediary between the user and the underlying data being manipulated (possibly multiple applications will work with the same data, but the principle remains the same). In general, the user need not be aware of how data are encoded and structured inside the application. Indeed, the very purpose of a good application is to shield the user from the ugly technical details. While it is conceivable that a technically astute person who is willing to invest the time and effort could decipher the internal structure of things, this would be an unusual thing to do as there is rarely much advantage to be gained. The purpose of the application itself is, after all, to make access to and manipulation of the data easier than digging around at the level of bits and bytes. There are exceptions to this, however. For example, most game programs deliberately impose obstacles on their players in order for play to be challenging. By tinkering around with the insides of such a program—dumping the data files and studying them, disassembling the program itself and possibly modifying it—it may be possible to "cheat." However, this sort of cheating has the flavor of cheating at solitaire: the consequences adhere to the cheater alone. There is a difference in that disassembling a game program is a puzzle-solving exercise in its own right, whereas cheating at solitaire is pointless, but the satisfactions to be gained from either, if any, are entirely personal.

If, however, a computer game involves multiple players, then delving into the program's internals can enable one to truly cheat, in the sense that one gains an unfair advantage over the other players, an advantage, moreover, of which they may be unaware. Habitat is such a multi-player game. When we were designing the software, our "prime directive" was, "The backend shall not assume the validity of anything a player computer tells it." This is because we needed to

protect ourselves against the possibility that a clever user had hacked around with his copy of the frontend program to add "custom features." For example, we could not implement any of the sort of "skill and action" elements found in traditional video games wherein dexterity with the joystick determines the outcome of, say, armed combat, because you could not guard against someone modifying their copy of the program to tell the backend that they had "hit," whether they actually had or not. Indeed, our partners at QuantumLink warned us of this very eventuality before we even started—they already had users who did this sort of thing with their regular system. Would anyone actually go to the trouble of disassembling and studying 100K or so of incredibly tight and bizarrely threaded 6502 machine code just to tinker? As it turns out, the answer is yes. People did. We were not 100% rigorous in following our own rule. It turned out that there were a few features whose implementation was greatly eased by breaking the rule in situations where, in our judgment, the consequences would not be material if some people "cheated" by hacking their own systems. Darned if some people didn't hack their systems to cheat in exactly these ways.

Care must be taken in the design of the world as well. One incident that occurred during our pilot test involved a small group of players exploiting a bug in our world database which they interpreted as a feature. First, some background. Avatars were hatched with 2,000 Tokens in their bank account and each day that they logged in they received another 100T. Avatars could acquire additional funds by engaging in business, winning contests, finding buried treasure, and so on. They could spend their Tokens on, among other things, various items for sale in vending machines called Vendroids. There were also Pawn Machines, which would buy objects back (at a discount, of course).

In order to make this automated economy a little more interesting, each Vendroid had its own prices for the items in it. This was so that we could have local price variation (i.e., a widget would cost a little less if you bought it at Jack's Place instead of The Emporium). It turned out that in two Vendroids across town from each other were two items for sale whose prices we had inadvertently set lower than what a Pawn Machine would buy them back for: Dolls (for sale at 75T, hock for 100T) and Crystal Balls (for sale at 18,000T, hock at 30,000T!). Naturally, a couple of people discovered this. One night they took all their money, walked to the Doll Vendroid, bought as many Dolls as they could, then took them across town and pawned them. By shuttling back and forth between the Doll Vendroid and the Pawn Shop for *hours*, they amassed sufficient funds to buy a Crystal Ball, whereupon they continued the process with Crystal Balls and a couple orders of magnitude higher cash flow. The final result was at least three Avatars with hundreds of thousands of Tokens each. We only discovered this the next morning when our daily database status report said that the money supply had quintupled overnight.

We assumed that the precipitous increase in "T1" was due to some sort of bug in the software. We were puzzled that no bug report had been submitted. By poking around a bit we discovered that a few people had suddenly acquired enormous bank balances. We sent Habitat mail to the two richest, inquiring as to where they had gotten all that money overnight. Their reply was, "We got it fair and square! And we're not going to tell you how!" After much abject pleading on our part they eventually did tell us, and we fixed the erroneous pricing. Fortunately, the whole scam turned out well, as the *nouveau riche* Avatars used their bulging bankrolls to underwrite a series of treasure hunt games which they conducted on their own initiative, much to the enjoyment of many other players on the system.

Keeping "Reality" Consistent

The urge to breach the boundary between the infrastructure level and the experiential level is not confined to the players. The system operators are also subject to this temptation, though their motivation is expediency in accomplishing their legitimate purposes rather than the gaining of illegitimate advantage. However, to the degree to which it is possible, we vigorously endorse the following principle:

- **Work within the system.**

Wherever possible, things that can be done within the framework of the experiential level should be. The result will be smoother operation and greater harmony among the user community. This admonition applies to both the technical and the sociological aspects of the system.

For example, with the players in control, the Habitat world would have grown much larger and more diverse than it did had we ourselves not been a technical bottleneck. All new region generation and feature implementation had to go through us, since there were no means for players to create new parts of the world on their own. Region creation was an esoteric technical specialty, requiring a plethora of obscure tools and a good working knowledge of the treacherous minefield of limitations imposed by the Commodore 64. It also required much behind-the-scenes activity of the sort that would probably spoil the illusion for many. One of the goals of a next generation Habitat-like system ought to be to permit far greater creative involvement by the participants without requiring them to ascend to full-fledged guru-hood to do so.

A further example of working within the system, this time in a social sense, is illustrated by the following experience. One of the more popular events in Habitat took place late in the test, the brainchild of one of the more active players who had recently become a QuantumLink employee. It was called the "Dungeon of Death." For weeks, ads appeared in Habitat's newspaper, *The Rant*, announcing that that Duo of Dread, "Death" and "The Shadow," were challenging all comers to enter their lair. Soon, on the outskirts of town, the entrance to a dungeon appeared. Out front was a sign reading, "Danger! Enter at your own risk!" Two system operators were logged in as "Death" and "The Shadow," armed with specially concocted guns that could kill in one shot, rather than the usual twelve. These two characters roamed the dungeon blasting away at anyone they encountered. They were also equipped with special magic wands that cured any damage done to them by other Avatars, so that they would not themselves be killed. To make things worse, the place was littered with cul-de-sacs, pathological connections between regions, and various other nasty and usually fatal features. It was clear that any explorer had better be prepared to "die" several times before mastering the dungeon. The rewards were pretty good: 1000 Tokens minimum and access to a special Vendroid that sold magic teleportation wands. Furthermore, given clear notice, players took the precaution of emptying their pockets before entering, so that the actual cost of getting "killed" was minimal.

One evening, one of us was given the chance to play the role of Death. When we logged in, we found him in one of the dead ends with four other Avatars who were trapped there. We started shooting, as did they. However, the last operator to run Death had not bothered to use his special wand to heal any accumulated damage, so the character of Death was suddenly and unexpectedly "killed" in the encounter. As we mentioned earlier, when an Avatar is killed, any

object in his hands is dropped on the ground. In this case, said object was the special kill-in-one-shot gun, which was immediately picked up by one of the regular players who then made off with it. This gun was not something that regular players were supposed to have. What should we do?

It turned out that this was not the first time this had happened. During the previous night's mayhem the special gun was similarly absconded with. In this case, the person playing Death was one of the regular system operators, who, accustomed to operating the regular Q-Link service, had simply ordered the player to give the gun back. The player considered that he had obtained the weapon as part of the normal course of the game and balked at this, whereupon the operator threatened to cancel the player's account and kick him off the system if he did not comply. The player gave the gun back, but was quite upset about the whole affair, as were many of his friends and associates on the system. Their world model had been painfully violated.

When it happened to us, we played the whole incident within the role of Death. We sent a message to the Avatar who had the gun, threatening to come and kill her if she did not give it back. She replied that all she had to do was stay in town and Death could not touch her (which was true, if we stayed within the system). OK, we figured, she's smart. We negotiated a deal whereby Death would ransom the gun for 10,000 Tokens. An elaborate arrangement was made to meet in the center of town to make the exchange, with a neutral third Avatar acting as an intermediary to ensure that neither party cheated. Of course, word got around and by the time of the exchange there were numerous spectators. We played the role of Death to the hilt, with lots of hokey melodramatic *shtick*. The event was a sensation. It was written up in the newspaper the next morning and was the talk of the town for days. The Avatar involved was left with a wonderful story about having cheated Death, we got the gun back, and everybody went away happy.

These two very different responses to an ordinary operational problem illustrate our point. Operating within the participants' world model produced a very satisfactory result. On the other hand, taking what seemed like the expedient course, which involved violating the world-model, provoked upset and dismay. Working within the system was clearly the preferred course in this case.

Current Status

As of this writing, the North American incarnation of Lucasfilm's Habitat, QuantumLink's "Club Caribe," has been operating for almost two years. It uses our original Commodore 64 frontend and a somewhat stripped-down version of our original Stratus backend software. Club Caribe now sustains a population of some 15,000 participants.

A technically more advanced version, called Fujitsu Habitat, has been operating for over a year in Japan, available on NIFTyServe. The initial frontend for this version is the new Fujitsu FM Towns personal computer, though ports to several other popular Japanese machines are planned. This version of the system benefits from the additional computational power and graphics capabilities of a newer platform, as well as the Towns' built-in CD-ROM for object imagery and sounds. However, the virtuality of the system is essentially unchanged and Fujitsu has not made significant alterations to the user interface or to any of the underlying concepts.

Future Directions

There are several directions in which this work can be extended. Most obvious is to implement the system on more advanced hardware, enabling a more sophisticated display. A number of extensions to the user interface also suggest themselves. However, the line of development most interesting to us is to expand on the idea of making the development and expansion of the world itself part of the users' sphere of control. There are two major research areas in this. Unfortunately, we can only touch on them briefly here.

The first area to investigate involves the elimination of the centralized backend. The backend is a communications and processing bottleneck that will not withstand growth above too large a size. While we can support tens of thousands of users with this model, it is not really feasible to support millions. Making the system fully distributed, however, requires solving a number of difficult problems. The most significant of these is the prevention of cheating. Obviously, the owner of the network node that implements some part of the world has an incentive to tilt things in his favor. We think that this problem can be addressed by secure operating system technologies based on public-key cryptographic techniques. (Rivest, Shamir and Adelman, 1978; Miller et al, 1987)

The second fertile area of investigation involves user configuration of the world itself. This requires finding ways to represent the design and creation of regions and objects as part of the underlying fantasy. Doing this will require changes to our conception of the world. In particular, we do not think it will be possible to conceal all of the underpinnings to those who work with them. However, all we really need to do is find abstractions for those underpinnings that fit into the fantasy itself. Though challenging, this is, in our opinion, eminently feasible.

Conclusions

We feel that the defining characteristic of cyberspace is the sharedness of the virtual environment and not the display technology used to transport users into that environment. Such a cyberspace is feasible today, if you can live without head-mounted displays and other expensive graphics hardware. Habitat serves as an existence proof of this contention.

It seems clear to us that an object-oriented world model is a key ingredient in any cyberspace implementation. We feel we have gained some insight into the data representation and communications needs of such a system. While we think that it may be premature to start establishing detailed technical standards for these things, it is time to begin the discussions that will lead to such standards in the future.

Finally, we have come to believe that the most significant challenge for cyberspace developers is to come to grips with the problems of world creation and management. While we have only made the first inroads onto these problems, a few things have become clear. The most important of these is that managing a cyberspace world is not like managing the world inside a single-user application or even a conventional online service. Instead, it is more like governing an actual nation. Cyberspace architects will benefit from study of the principles of sociology and economics as much as from the principles of computer science. We advocate an agoric, evolutionary approach to world building rather than a centralized, socialistic one.

We would like to conclude with a final, if ironical, admonition, one that we hope will not be seen as overly contentious:

- **Get real.**

In a discussion of cyberspace on Usenet, one worker in the field dismissed Club Caribe (Habitat's current incarnation) as uninteresting, with a comment to the effect that most of the activity consisted of inane and trivial conversation. Indeed, the observation was largely correct. However, we hope some of the anecdotes recounted above will give some indication that more is going on than those inane and trivial conversations might indicate. Further, to dismiss the system on this basis is to dismiss the users themselves. They are paying money for this service. They do not view what they do as inane and trivial, or they would not do it. To insist this presumes that one knows better than they what they should be doing. Such presumption is another manifestation of the omniscient central planner who dictates all that happens, a role that this entire article is trying to deflect you from seeking. In a real system that is going to be used by real people, it is a mistake to assume that the users will all undertake the sorts of noble and sublime activities which you created the system to enable. Most of them will not. Cyberspace may indeed change humanity, but only if it begins with humanity as it really is.

Notes

1: One of the questions we are asked most frequently is, "Why the Commodore 64?" Many people somehow get the impression that this was a technical decision, but the real explanation has to do with business, not technology. Habitat was initially developed by Lucasfilm as commercial product for QuantumLink, an online service (then) exclusively for owners of the Commodore 64. At the time we started (1985), the Commodore 64 was the mainstay of the recreational computing market. Since then it has declined dramatically in both its commercial and technical significance. However, when we began the project, we did not get a choice of platforms. The nature of the deal was such that both the Commodore 64 for the frontend and the existing QuantumLink host system (a brace of Stratus fault-tolerant minicomputers) for the backend were givens.

2: Habitat contains its own fully-fledged economy, with money, banks, and so on. Habitat's unit of currency is the Token, reflecting the fact that it is a token economy and to acknowledge the long and honorable association between tokens and video games. Incidentally, the Habitat Token is a 23-sided plastic coin slightly larger than an American quarter, with a portrait of Vernor Vinge and the motto "*Fiat Lucre*" on its face, and the text "Good for one fare" on the back; these details are difficult to make out on the Commodore 64 screen.

Acknowledgements

We would like to acknowledge the contributions of some of the many people who helped make Habitat possible. At Lucasfilm, Aric Wilmunder wrote much of the Commodore 64 frontend software; Ron Gilbert, Charlie Kelner, and Noah Falstein also provided invaluable programming and design support; Gary Winnick and Ken Macklin were responsible for all the artwork; Chris Grigg did the sounds; Steve Arnold provided outstanding management support; and George Lucas gave us the freedom to undertake a project that for all he knew was both impossible and insane. At Quantum, Janet Hunter wrote the guts of the backend; Ken Huntsman and Mike Ficco provided valuable assistance with communications protocols. Kazuo Fukuda and his crew at Fujitsu have carried our vision of Habitat to Japan and made it their own. Phil Salin, our boss at AMiX, let us steal the time to write this paper and even paid for us to attend the First Conference

on Cyberspace, even though its immediate relevance to our present business may have seemed a bit obscure at the time. We would also like to thank Michael Benedikt, Don Fussell, and their cohorts for organizing the Conference and thereby prompting us to start putting our thoughts and experiences in writing.

Bibliography

- Alber, A. F. (1985). *Videotex/Teletext: Principles and Practices*. NY: McGraw-Hill, .
- American National Standards Institute. (1983, December). *Videotex/Teletext Presentation Level Protocol Syntax*. North American PLPS.
- Drexler, K. E.. (1986). *Engines of Creation*. Garden City, NY: Anchor Press.
- Drexler, K. E. & Miller, Mark S. (1988). Incentive Engineering for Computational Resource Management. In Huberman, B.A. (Ed.). *The Ecology of Computation*. Amsterdam: Elsevier Science Publishers, p. 231-266.
- Gibson, W. (1984). *Neuromancer*. NY: Ace Books.
- Goldberg, A. & Robson, D. (1983). *Smalltalk-80: The Language and Its Implementation*, Reading, MA: Addison-Wesley.
- Hayek, F. A. (1973). *Law Legislation and Liberty, Volume I: Rules and Order*, Chicago: University of Chicago Press.
- (1978). *New Studies in Philosophy, Politics, Economics, and the History of Ideas*. Chicago: University of Chicago Press.
- (1989). *The Fatal Conceit*. Chicago: University of Chicago Press.
- International Standards Organization. (1986, June). *Information Processing Systems -- Open System Interconnection -- Transport Service Definition*. International Standard number 8072.
- Miller, M. S.; Bobrow, D. G.; Tribble, E. D.; & Levy, D. J. (1987). Logical Secrets. In Shapiro, E. (Ed.). *Concurrent Prolog: Collected Papers*. Cambridge, MA: MIT Press.
- Miller, M. S., & Drexler, K. E. (1988a) Comparative Ecology: A Computational Perspective. In Huberman, B.A. (Ed.). *The Ecology of Computation*. Amsterdam: Elsevier Science Publishers.
- (1988b). Markets and Computation: Agoric Open Systems. In Huberman, B.A. (Ed.). *The Ecology of Computation*. Amsterdam: Elsevier Science Publishers.
- Popper, K. R. (1971/1962). *The Open Society and Its Enemies*. Princeton, NJ: Princeton University Press.
- (1972). *Objective Knowledge: An Evolutionary Approach*, (Oxford: Oxford University Press.
- Rivest, R.; Shamir, A., & Adelman, L. (1978, February). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM 21*, p. 120-126.
- Sowell, T. (1987). *A Conflict of Visions*. NY: William Morrow.
- Sterling, B. (Ed.). *Mirrorshades: The Cyberpunk Anthology*. NY: Arbor House.
- Sussman, G. J. & Abelson, H. (1985). *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press.
- Vinge, V. (1981). True Names. In Vinge, Vernor. *Binary Star #5*. NY: Dell Publishing Company.