Implementing Gibsonian Virtual Environments

Dieter Schmalstieg and Michael Gervautz
Institute of Computer Graphics
Vienna University of Technology, Austria
email: schmalstieg|gervautz@cg.tuwien.ac.at

Abstract. We speculate on the possibilities for a realization of the idea of a three-dimensional Cyberspace, as sketched in the novels of William Gibson using today's technology. Virtual reality and global networks such as the Internet together can provide the fundamentals for such an effort. From an examination of the technological principles of a Gibsonian virtual environment, we conclude what properties such an implementation must have. An implementation of a system with these properties faces a number of problems in the large and in the small; especially, the semantics of the virtual environment and its inhabitants must be defined. We also report on our ongoing implementation of a prototype software system which was designed to evaluate the feasibility of these ideas.

1. Introduction

In his famous novel "Neuromancer" [Gibs84], William Gibson presents the idea of the Cyberspace, a sensual representation of a global, networked database that is presented to the user directly by brain stimulation. While such technology remains fiction, recent trends indicate that the fundamental idea of Cyberspace is not only a vision for technologists, but there may be ways to implement it. A possible implementation will depend on technology from two major areas:

- Virtual reality research has provided means of conveying multi-sensory information to human participants in real time, with the goal of creating the illusion of presence in a simulated environment.
- Global networks are expanding at a dramatic pace. Without exaggeration it may be stated that the way we communicate is being revolutionized.

It is important to realize that while Gibsonian Cyberspace is a fictional concept, it serves as a source of inspiration for the development of virtual reality technology. If we take Gibson literally, we must separate those concepts from his fictitious world that can be realized from those that cannot (yet?). Once this is done, strategies for implementing the possible can be developed. Gelernter has shown that a structured approach to such a venture is possible [Gele92].

A good starting point for such a plan is to analyze the most important network infrastructure of these days - the World Wide Web. It has proven to be operational on a truly global scale, and it's popularity is overwhelming. We believe that its success can be attributed to a unique combination of factors, among them the simplicity of the hypertext paradigm, platform independence via the use of standardized protocols, and cost-effectiveness by running on garden-variety PCs.

However, we should realize that the WWW ingeniously circumvents most of the hard problems of large distributed databases: The file-oriented server architecture makes persistence issues trivial. Generally unrestricted access and connection-less client-server protocols largely avoid the need for synchronization. A relatively strict separation of information producers and information consumers effectively makes information flow one way only, thereby working around consistency issues. The insignificance of synchronization persistence, in combination with the weak “glue” of hyperlinks and URLs that connect local information spaces, allows simple and unbound scalability - up to a literally global size. The nature of interaction (document retrieval) does not impose any hard limits on responsiveness - people are willing to live with bad performance (“World Wide Wait”).

These issues that the WWW so cleverly avoids *must* be addresses when attempting to move towards Gibsonian virtual environments. Therefore we cannot simply extend the WWW or any other current network infrastructure to three dimensions. New ideas are needed.

The goal of this paper is to gain insight in the feasibility of a realization of Cyberspace. We therefore first examine the technical side of Gibson’s fiction (section 2), and analyze the state of current technology (section 3). We then discuss important semantic questions and design considerations for a possible implementation of a Gibsonian virtual environment. An analysis of this kind has to operate on multiple layers of detail. Some considerations apply to the environment as a whole (macroscopic level), others to the elements that form the environment (microscopic level). On each level, issues can be according to the underlying relation: Questions concerning semantics and design of single components (intrarelation), and questions concerning the interaction (interrelation) between multiple components. The table below shows some of the questions, and also serves as a map to the rest of this paper.

Questions	**Macroscopic**	**Microscopic**
Intrarelation	How does the environment relate to itself (in the presence of multiple users)? What are the semantics of shared content? (section 4.1)	What is the structure of the elements that the virtual environment is composed of? (section 5.1)
Interrelation	How do multiple environments relate to each other? How are environments connected? (section 4.2)	How do the elements of the virtual environment interact? How can objects live in different virtual environments? (section 5.2)

Finally, we report on our ongoing prototype implementation of a Gibsonian virtual environment (section 6).

2. Concepts of Gibsonian Cyberspace

In the real world, we perform interaction in three-dimensional space. The largest amount of information is input to the human via the visual system, and our manual capabilities are very sophisticated. Consequently, *three-dimensional interaction* is the most natural form of information exchange for human beings. Therefore an artificial system that successfully mimics the way interaction takes place in real life will be very effective. This does not mean that the real world should be duplicated to the most minute detail at all cost. Large amounts of information have to represented by abstractions to be usable, but clearly three-dimensional abstractions are more powerful than conventional ones. The problem so far with this kind of paradigm is insuperior quality: Crude and slow three-dimensional applications are not accepted, but technological advances will address this problem. A workable implementation of Cyberspace implies that the *performance* problem for typical interactions is generally solved (be it by powerful hardware or by sophisticated software), so that smooth interaction in near real-time is available.

A global information space should be *continuous*. By this we mean the user’s view of the information should not be restricted by arbitrary bounds in the domain of the metaphor. For hypertext systems such as the WWW this means that links work the same way no matter whether the target document is local or remote. For three-dimensional virtual environments, this means that information is arranged spatially, and that the world does not “end” arbitrarily. With continuity comes the need for scalability: A global information space is only workable if the architecture is easily scaleable.

A continuous information space creates the expectation that the information is persistent. Things in the real world generally do not vanish arbitrarily or change in unpredictable manner. Therefore such a behavior is undesirable, because it is confusing for the user. This requires that information entities have some form of stable state, that can be accessed repeatedly and deterministically. Such a constraint is definitely hard to fulfill, especially if the state is subject to changes due to ongoing simulation, and requires safe mechanisms similar to database transactions, that conflict with real-time responsiveness. If the underlying software architecture operated with some form of data replication, consistency and synchronization issues must also be solved.

Pure information access is a form of interaction with software agents (be they intelligent or not). However, a virtual environment also raises the desire for *interaction between multiple human users*. If the virtual environment can become the medium for communication between users, very promising applications can be realized. Beyond multi-user interaction, the content of the virtual environment should be diverse. In principle, everyone should be able to contribute to the environment with reasonable effort (i.e. populate it with data, services, agents etc.). People will only be willing to contribute to a sufficiently *democratic* information infrastructure. This requires that on the one hand standards are used so that different components can talk to each other, on the other hand that the system is open so that novel features or implementation of accepted features can be made operational.

3. Current state of networked information spaces

In order to define a status quo, we will briefly elaborate on today’s virtual environments, which are still largely research prototypes. Virtual environments such as DIVE [Carl93] and MR [Shaw93]/EM [Wang95], that allow multiple user s to interact with each other and with autonomous agents in a simulated 3-D space, are usually limited to a small number of users and run on local networks for performance reasons. Typically applications are some form of computer supported cooperative work (design, information sharing, training, playing).

Examples for systems that scale beyond local workgroup usage are rare. The most prominent representative is the NPSNET [Mace94] family: It has been demonstrated to work for a large number of concurrent users, but it is limited to a specific domain (military simulation and training), and the underlying networking protocol serverly limits the variety of possible interactions.

These research systems provide a framework from which arbitrary applications can be built, but they generally fail to provide a solution for data persistence. Typically, application specific data is loaded at system start, and the internal state of the system changes as the simulation evolves. However, dynamic reconfiguration of applications is limited, and often the internal state of the system has to be abandoned for such a task.

4. Macroscopic considerations

4.1 Semantics for shared virtual environments

Virtual environments, as has been indicated, should support multiple concurrent users. This produces some subtle semantic problems. If multiple users share one virtual environment, it is not necessary that the same content is presented to every user. Furthermore, if some or all of the content is presented to multiple users, any changes one user makes can or cannot affect the view others have of the manipulated environment. Semantic rules are required that define the way the data is managed by the system. In the following, we present four different semantic models, from the simplest to the most complicated, which is true to Gibson's spirit, but also most difficult to implement.

- Sharing is relatively simple if no modifications to the content of the virtual environment are allowed. This is frequently used for so-called walkthrough applications with the aim of data visualization (e.g. architectural models). The user cannot manipulate anything, so there is not need to communicate anything about the user to other participants in the same environment. This scheme is used by the current Virtual Reality Modeling Language [Ball95] extension to the WWW. A scene (environment?) is downloaded upon request from a WWW server, and then displayed locally. Multiple users can view the same scene concurrently, but no real sharing occurs.
- As an extension to the pure walkthrough model, objects in the environment can be attributed with behavior [Zyda93]. The user can interact with a local instance of the environment and make modifications. However, these changes are not propagated to other users present in the same environment, and changes are not recorded anywhere.
- The previous model can be extended if changes can be propagated between users that share one virtual environment. This requires a communication channel between the users - either directly in a peer-to-peer style, or mediated by a server. It has to be defined which set of users are sharing the same virtual environment (compare [Sing94]). A user can join the environment by logging into a session. Multiple instances of the same environment can exist concurrently, with different sets of users and different internal state. A new session can be started at any time, with default values for all changeable attributes. Thus there is no persistence. Such a model is useful for multi-user games that require sharing of the gaming environment and state, but also the possibility to reset the environment for a new game.
  The sharing in such an environment makes it necessary to solve consistency problems [Blau92]. Consider two users that make concurrent updates to their respective local copies of an object, and then propagate the changes. A priority mechanism is needed that resolves such conflictual situations (often based on timestamps).
  Scaling of such a shared environments is compromised by a high communication cost that comes from the need to propagate changes to all participants. This problem can be partly addressed by limiting propagation of messages to the subset of participant that really need to receive the message, typically defined by

geometric proximity (cf. [Mace95] or [Fahl93], and by local simulation of remote activity (e.g. [Sing95]).

- Finally, a shared multi-user environment can be equipped with support for persistence. If a user makes a change to the environment, the change is not only propagated to other users that are in the environment at the same time, but also permanently recorded. A user entering the environment later will see all changes made before his or her advent. Conceptually, there are no longer multiple local copies of one environment, but only a single distributed environment. This makes an implementation hard for pure peer-to-peer network architectures that are otherwise often preferred for performance reasons: If the last user leaves the environment, knowledge about the changes is gone.
  However, even if a central authority (server) is available, it is not trivial to record the current state of a "live" virtual environment: A log of changes can be built, but it may be difficult to reconstruct everything from the log. The other option, a "snapshot" of the dynamic state of the environment at regular intervals may be infeasible because of the size of the underlying database. While this is obviously a difficult problem, it is mandatory to provide a solution in order to create a persistent global Cyberspace: It cannot be expected that all parts of the system operate without errors or down-times. Therefore only fault-tolerant operation with degraded performance (some region of the information space is unavailable) combined with error recovery (that allows reconstruction of a recent dynamic state of the environment) can provide a workable solution.

Clearly, a shared persistent environment is the hardest to implement. However, it is also the only model the semantics of which are clearly Gibsonian. While the other models obviously have useful applications, they do not embody the role of Cyberspace as the medium, not only the way of presentation of content. It is therefore important to aim at implementing both sharing and persistence.

4.2 Connecting virtual environments

Once it is clear how a single virtual environment works, we can try to answer the question what the relation of two or more virtual environments is. The question is, where does the participant go when leaving one environment: Either back to the real world, or to another virtual environment. The way by which virtual environments are interconnected have a strong influence on how the virtual environment is perceived, and what human users expect from it. There are two models how virtual environments can coexist:

1. The dimension of the virtual environment has been defined to be three-dimensional space. One can view the environment itself to be of the same dimension as the contained elements. In this case, the environment occupies a 3-D region. Two neighboring environments can share a common border, in other words, they have a spatial relation.
2. The environment has a higher dimension than the contained information. Two environments can therefore not share a common border, they can only coexists in parallel dimensions.

The first metaphor resembles our perception of places that can be reached by spatial traversal (walking, flying, ...). While this is appealing because of its simplicity, it requires that the information space is subdivided among environments (read: sites, nodes, servers, ...) [Pesc95]. We can expect that the same problems will arise for Cyberspace real estate, that we know from the real world.

The alternative - multiple co-dimensions - does not allow a participant to use the same metaphor for the traversal of multiple environments that is used for traversing within an environment. Instead, a dimensional shift between environments has to be provided, the 3-D equivalent of a hyperlink (*portals*). The creator of an environment has complete freedom to use all of 3-D information space. However, it is unlikely that it will be filled with meaningful content. We can rather expect that only a small isolated island will be present, surrounded by a vast void, which might be disturbing for users.

The 3-D spatial arrangement appears natural and human-centric, but that does not necessarily mean that real-world geometric relations have to be maintained at all costs. A virtual environment should be able to represent situations that are not possible in a real environment. Here we give some examples:

- A continuous space can be equipped with virtual "wormholes" that allow a user to teleport to an arbitrary place. This does not break the metaphor and adds additional value.
- The Euclidean geometry that is fundamental to 3-D space can also be extended in non-standard ways, which may provide interesting applications. Environments could be connected in loops, potentially Moebius or other strange loops. Spatial arrangement of environments may be done using twists and warps (e.g. non-uniform scaling of regions). Finally, Non-Euclidean geometry can be used to provide totally unseen experiences (compare [Gröl95], [Gröl94], [Geom93]).

From the point of view of implementation, model 2 (hyperlinked virtual environments) are definitely easier to handle. Every environment creator has all of 3-D space available, and does not need to negotiate with other creators what region to use. It would also avoid wrong expectations about a correspondence between real distances (between servers) and virtual distances (in information space). However, as we have seen the continuous 3-D space is the more natural and more flexible metaphor. It is also the only option in the spirit of Gibsonian Cyberspace. Implementational difficulties should therefore be considered a challenge to be overcome.

5. Microscopic considerations

5.1 Object architecture for virtual environments

With the semantics of the environment itself have being defined, we can risk a deeper look into the architecture of the environment. From an implementational point of view, the virtual environment is a (distributed) database. The database is composed of items that go by different names, such as object, artifact, entity, actor, ... For this discussion, we will adopt the generic term object. Note that although concrete virtual environment implementations are mostly object-oriented, this is not a requirement.

The software architecture underlying the objects that form the virtual environment determines the implementation of most other parts of the software system, and must therefore be carefully considered.

- **Passive objects**: The simplest architecture defines completely passive objects as simple data containers. Often the objects is simply defined by its geometric representation (e.g. VRML 1.0). Every modification to an object is being executed by the environment in which the object lives. Because it can only respond to external stimulus, the object's behavior is completely deterministic. The functions that can be invoked on the object are determined by the system's set of features, that is typically not easily extended. While the implementation of such a structure is relatively simple, the absence of a modular design makes it hard to extend the system, which compromises generality.
- **Active objects with deterministic behavior**: Object representation can be extended by simple built-in behavior that is composed from standard mechanisms,

such as kinematic animation and standard interaction (e.g. picking, dragging, scaling). A good example for such capabilites is OpenInventor. Note that while objects in this class are more active than in the previous class, they are still deterministic, i.e. their behavior is predictable.

- **Active objects with non-deterministic behavior**: Objects for which behavior can be specified by the creator, and that can hold internal state, can be equipped with nondeterministic behavior. Typically, some form of scripting language is used to formulate the object's behavior - if the scripting language is powerful enough, interesting nondeterminstic behavior can be created (formally, the language has to be Turing-complete; practically, it has provide enough features so that programming does not require too much effort).
- **Active objects with external stimulus**: If the built-in capabilities of the environment - even in combination with user programming - are not powerful enough to allow the construction of the desired application (e.g. because of performance reasons), an interface to external applications is required. Any application can control objects in the environment, as long as it complies to the application interface [Hond95]. The source of the external control should not be visible to the user in the environment, communication with the external application should be mediated via the object. From the environments point of view, the user is an external application which is in control of the user's representation in the environment (or Avatar).

Obviously, all of these classes of objects are needed to provide the power for the environment to support any kind of application. Object-oriented system design helps to hide implementational differences from the users.

5.2 Object semantics in multiple virtual environments

Unfortunately, defining the architecture of objects in virtual environments is not sufficient to explain everything semantically important for the objects. We still have to explain how objects relate to each other, and how objects relate to (multiple, diverse) environments.

The relationship between objects in a virtual environment is defined by the objects' possibilities to communicate. This is most easily and flexible achieved by a generic message passing mechanism that allows arbitrary messages to be sent between objects. The progress that makes the virtual environment come alive is then carried by the distributed simulation in the form of objects sending each other messages and responding to them. We will not discuss technical details of message passing mechanism that would break the scope of what is possible here. Instead, we concentrate on the interpretation of messages: How can objects understand each other?

- *How do objects understand messages*? There is no universal message language. The meaning of a message is defined by convention. For an evolving system, that means that messages defined at a later time will not be understood by objects created at an earlier time, simply because the meaning of the message was not defined then. Therefore, a mechanism is needed that that allows interpretation of a message. An oracle provides a solution [Snow94], but it requires a central authority that may be hard to set up in a distributed system. An alternative is to arm objects with learning skills, but this is again a difficult problem for which no general solution exists.
- *How do objects survive in worlds with different requirements*? Up to this point, we have silently assumed that virtual environments only vary in content (i.e. in the objects they contain), not in their fundamental setup. Actually, a global network of environments requires a set of standards that are kept by all sites, or the system

will fail to work. However, that does not mean that there cannot be variation in the conditions of the surrounding provided by the environment to its inhabitating objects. The problem does not arise if only objects that are designed to live in a particular type of environment are allowed. But what happens if we allow objects (or users) to migrate between worlds?

For example, a world may define gravity and other physical laws to work on the contained objects. How can an object that does not know about physics behave reasonable in this environment? And vice versa: How can an object that requires physical laws for fundamental operations such as movement to work, exist in an environment without such properties? The general answer is, if the object is capable of adapting autonomously to the situation, the problem can be solved. A method for this has been proposed in [Snow94], but is unclear if it is generally workable.

We favor an alternative approach: We define all requirements not on a per-environment base, but rather by having separate groups for every environmental influence. Membership in such a group means that the object-environment communication regarding this matter is carried out. The object itself or a local authority decides whether membership in a particular group is appropriate or not. This leaves the requirement that the object is capable of very simple fundamental actions, such as movement in 3-D or the ability to display itself.

6. A first step

We are currently implementing a distributed virtual environment infrastructure that we hope will help us to explore the possibilities of a Gibsonian Cyberspace. The environment is fueled by a network of servers. The servers divide the world among each other into regions that are connected geometrically (currently we are using a regular grid). Interest in the objects is local, so that the environment software only has to maintain communication channels with their immediate neighbors (in information space). This is important for slow network connections, because it puts a natural limit on the amount of network communication.

While the servers provide a space for the objects to live, and takes care of the ongoing simulation, users can connect to a server to be present in the environment. A network connection is set up between environment server and user client, and information about the environment’s content, and changes, are transmitted to the client as needed. In particular, we are developing an optimized protocol for remote display of 3-D graphics, based on VRML (high performance 3-D graphics is very important for the immersive quality of the environment). Using the client software, the user is able to inspect the environment, and also to interact with it. When a user leaves the region of one server and enters the domain of another, the network connection is transiently passed on.

Some users may want to contribute to the virtual environment. This can be done by the creation of new objects, or by the creation of new object types. The behavior of objects can be specified using an object-oriented scripting language. Additionally, an object can be configured to be controlled by an external application, so that arbitrary software systems can be integrated.

7. Conclusions

We have discussed the possibilities of implementing three-dimensional virtual environments in a large networked infrastructure. Our concept is inspired by the idea of Cyberspace in the Gibsonian sense. Such a world-wide virtual environment will have to be three-dimensional, highly interactive, spatially continuous, multi-user capable, shared, persistent, scaleable and democratic. Clearly, a system with such demanding properties cannot easily be extrapolated from existing network paradigm such as the WWW. A new concept is needed, that provides solution to design issues and semantic questions. For the environment as a whole, it is important what sharing of the content between multiple users means and how multiple worlds can be connected. For the inhabitants or objects of the environment we have suggested how they can be structured, how they communicate, and how they can exist in multiple virtual environments under diverse conditions. Once logically sound and technically feasible answers to these problems have been provided, an implementation of Gibsonian virtual environments should become (virtual?) reality.

Acknowledgements. The authors would like to thank Peter Stieglecker for contributing essential ideas to this work, and for coming up with the original idea of a continuous Cyberspace in the first place. The first author was supported by a stipend from the faculty of technical and natural sciences of the Vienna university of technology.

References

[Ball95] G. Ball, A. Parisi, M. Pesce: VRML Draft Specification 1.0. Technical Report, http://www.eit.com/vrml/vrmlspec.html (1995)

[Blau92] Blau B., Hughes C. E., Moshell J. M., Lisle C.: Networked virtual environments. SIGGRAPH Symposium on Interactive 3D Graphics, Vol. 25, No. 2, pp. 157-160 (1992)

[Carl93] C. Carlsson, O. Hagsand: DIVE- A platform for multi-user virtual environments. Computers & Graphics, Vol. 17, No. 6, pp. 663-669 (1993)

[Fahl93] L. Fahlen, O. Brown, C. Carlsson: A Space Based Model for User Interaction in Shared Synthetic Environments. Proceedings of InterCHI, pp. 43-48 (1993)

[Gele92] D. Gelernter: Mirror Worlds. Oxford University Press (1992)

[Geom93] Geometry Center, University of Minnesota: Not Knot. Animation film, SIGGRAPH'93 (1993)

[Gröl94] E. Gröller, H. Löffelmann: Extended Camera Specification for Image Synthesis. Machine Graphics and Vision, Vol. 3, No. 3, pp. 514-530 (1994)

[Gröl95] E. Gröller: Nonlinear Ray-Tracing: Visualizing Strange Worlds. Visual Computer, Vol. 11, No. 5, pp. 263-274 (1995)

[Hond95] Y. Honda, K. Matsuda, J. Rekimoto, R. Lea: Virtual Society: Extending the WWW to support a Multiuser Interactive Shared 3D Environment. Proc. of VRML'95 Symposium, San Diego (1995)

[Mace94] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, S. Zeswitz: NPSNET: A Network Software Architecture for Large-Scale Virtual Environment. Presence, Vol. 3, No. 4, pp. 265-287 (1994)

[Mace95] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, P. Barham: Exploiting Reality with Multicast Groups: A Network Architecture for Large-scale Virtual Environments. Proceedings of VRAIS'95 (1995)

[Pesc95] M. Pesce, P. Kennard, A. Parisi: Cyberspace. Technical report, http://vrml.wired.com/concepts/pesce-www.html (1995)

[Sing94] G. Singh, L. Serra, W. Png, Hern Ng: BrickNet: A Software Toolkit for Network-Based Virtual Worlds. Presence, Vol. 3, No. 1, pp. 19-34 (1994)

[Sing95] S. Singhal, D. Cheriton: Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality. Presence, Vol. 4, No. 2, pp. 169-194 (1995)

[Shaw93b] C. Shaw, M. Green, J. Liang, Y. Sun: Decoupled simulation in virtual reality with the MR toolkit. ACM Transactions on Information Systems, Vol. 11, No. 3, pp. 287-317 (1993)

[Snow94] D. Snowdon, A. West: AVIARY: Design Issues for Future Large-Scale Virtual Environments. Presence, Vol. 3, No. 4, pp. 288-308 (1994)

[Wang95] Q. Wang, M. Green, C. Shaw: EM - An Environment Manager for Building Networked Virtual Environments . Proceedings of VRAIS'95 (1995)

[Zyda93] M. Zyda, D. Pratt, J. Falby, C. Lombardo, K. Kelleher: The Software Requred for the Computer Generation of Virtual Requirements. Presence, Vol. 2, No. 2, pp. 131-140 (1993)