HAL open science

Solipsis: A Decentralized Architecture for Virtual Environments

Davide Frey, Jérôme Royan, Romain Piegay, Anne-Marie Kermarrec, Emmanuelle Anceaume, Fabrice Le Fessant

To cite this version:

Davide Frey, Jérôme Royan, Romain Piegay, Anne-Marie Kermarrec, Emmanuelle Anceaume, et al.. Solipsis: A Decentralized Architecture for Virtual Environments. 1st International Workshop on Massively Multiuser Virtual Environments, Mar 2008, Reno, NV, United States. inria-00337057

HAL Id: inria-00337057

https://inria.hal.science/inria-00337057

Submitted on 5 Nov 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Solipsis: A Decentralized Architecture for Virtual Environments

D. Frey* INRIA Rennes
J. Royan† Orange Labs
R. Piegay ‡ Orange Labs
A.-M. Kermarrec § INRIA Rennes
E. Anceaume ¶ IRISA Rennes
F. Le Fessant ‖ INRIA Paris

Abstract

Lack of scalability is a key issue for virtual-environment technology, and more generally for any large-scale online experience because it prevents the emergence of a truly massive virtual-world infrastructure (Metaverse). The Solipsis project tackles this issue through the use of peer-to-peer technology, and makes it possible to build and manage a world-scale Metaverse in a truly distributed manner. Following a peer-to-peer scheme, entities collaborate to build up a common set of virtual worlds. In this paper, we present a first draft of the Solipsis architecture as well as the communication protocol used to share data between peers. The protocol is based on Raynet, an n-dimensional Voronoi-based overlay network. Its data-dissemination policy takes advantage of the view-depedent representation of 3D contents. Moreover, the protocol effectively distributes the execution of computationally intensive tasks that are usually executed on the server-side, such as collision detection and physics computation. Finally, we also present our web component, a 3D navigator that can easily run on terminals with scarce resources, and that provides solutions for smooth transitions between 3D Web and Web 2.0.

Keywords: Peer-to-peer System, Metaverse, Shared Virtual Worlds, Massively Decentralized System, Adaptative 3D Streaming.

Index Terms: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

1 Introduction

The Metaverse concept, first described by Neal Stephenson in his science fiction novel 'Snow Crash' published in 1992, and more generally depicted in the whole cyberpunk writing movement, has deeply influenced generations of virtual reality pioneers, artists, game designers and nowadays virtual worlds enthusiasts. Over the last sixteen years, the notion has evolved toward a synonym for virtual world, loosing progressively the 'Universe' part of the concatenation and the huge and infiniteness feeling emanating from it.

However, for us the Metaverse is a system of numerous, interconnected virtual and typically user-generated worlds (or Metaworlds) all accessible through a single-user interface. According to this strict definition, the only Metaverse existing today is a prehistoric one: the World Wide Web itself. Plenty of virtual worlds flourish these days claiming they are the Metaverse, but they only are a part of it, as websites are the leaves of the worldwide web tree. We need three things to reach this cyberpunk authors' dream. First, a way to sustain the incredible amount of data and MIPS involved, then a set of protocols to provide interoperability and finally new tools to build virtual worlds as easily as a traditional HTML page. These requirements are the cornerstones of our project.

The paper is structured as follows. We first present a synthetic overview of the Solipsis research project in Section 2, and an analysis of related work in Section 3. Then we describe how we envision to manage decentralized virtual worlds, describing in details our peer-to-peer architecture, how we manage 3D-model sharing and physics computation, and how to stream 3D contents in an adaptative way. Finally, in Section 5, we present our navigator, which is the user interface used to interact with the Solipsis Metaverse, while in Section 6, we conclude the paper and outline our future directions.

2 Project overview

In a word, Solipsis is a platform for massively multi-participant and user-generated virtual worlds. It relies on a peer-to-peer architecture that makes scalability its main characteristic: the universe may thus be inhabited by an unlimited number of participants. As there is no central authority, the virtual universe is by definition public and the inhabitants' freedom, as well as the world builders and developers' imagination, are boundless.

2.1 Expected results

We seek to spark off the emergence of an unbounded public virtual universe that is designed, created, run, but also potentially hosted, by people throughout the world. Freedom can be spotted as the main characteristic of this opensourced system (license GNU/GPL v2+) because the virtual universe does not belong to any organization; it belongs to all the users.

The major deliverable is a new network communication protocol adapted to the strong constraints of self-produced environments, to massively multi-user applications and to virtual reality, which make the system more scalable as the resources its uses are those provided by its users.

We also work on creating an ergonomic and user-friendly interface to allow non-professional agents to easily create 3D scenes and contents (declarative or automatic modeling, tagging..). Nevertheless, we do not want to mark a break with the actual flat 2D web; rather, we aim to start a smooth transition towards an immersive Internet making the most of both 2D and 3D. As we will see in part V, our navigator can be embedded in a regular webpage - in-web world - or map regular webpages as an interactive texture on any 3D surface - in-world web.

The last major expected result is a deep analysis on the behavior (virtual social life, game, education, services..) of metaverse users to give directions for future developments.

3 Related Work

The Web3D consortium originally had the ambition to create a freely navigable online world [19]. Unfortunately, due to technical constraints, such as bandwidth limitations, the VRML standard was only used to encode simple 3D contents in order to visualize them on web sites. Only related research and bandwidth increase have now made it possible to draft the architecture of a Web3D, metaverse or cyberspace.

*e-mail: davide.frey@irisa.fr
†e-mail: jerome.royan@orange-ftgroup.com
‡e-mail: romain.piegay@orange-ftgroup.com
§e-mail: Anne-Marie.Kermarrec@irisa.fr
¶e-mail: emmanuelle.anceaume@irisa.fr
‖e-mail: fabrice.le_fessant@inria.fr

3.1 Compression and adaptive 3D Streaming

A lot of research work has focused on the compression of 3D models, as well as on adaptive streaming methods that allow a progressive and fast transmission over networks of the required 3D contents visualized from the current viewpoint.

First, existing 3D compression algorithms use both techniques adapted from the 1D and 2D cases (like wavelets, entropy coding, and predictive coding), and completely different approaches that take advantage of the properties of 3D surfaces (like Edgebreaker, Subdivision Surfaces, and triangle strips)[9][13]. Also, parametric solutions, that provide intuitive modeling tools, are widely used to create avatars, complex creatures, or vehicles in games. More complex 3D contents can be created using procedural modeling solutions that have the drawback of requiring a reconstruction process executed on the fly on the client side. Parametric and procedural modeling provide very good compression rate compared to usual mesh compression algorithms, but no generic solutions exist to model a great variety of objects.

The second solution consists in filtering the 3D contents required to visualize the scene from a given viewpoint. Indeed, a complete download of a huge 3D scene is not necessary to render with optimal details the virtual environment from a given viewpoint. First, the area of a 3D model projected on the screen depends not only on its size, but especially on its distance from the viewpoint. Many solutions have therefore been proposed to adapt the resolution of 3D models to the current viewpoint [8]. Continuous levels of detail allow to transmit refinements progressively as the viewpoint comes closer to 3D objects [11, 17]. Second, most of 3D objects in huge and complex 3D scenes are occulted by other ones during the navigation. Thanks to an offline computation of 3D objects visible from regions resulting from a partitioning of the navigation area, servers can signifanctly reduce the amount of data that has to be sent to the client, without affecting the visual quality [21, 22]. Unfortunately, visibility filtering methods have to be disabled during flying-over navigation, since occultations are clearly limited. In fact, these filtering methods provide multi-resolution functionalities allowing an adaptive 3D streaming of the virtual environments.

3.2 Centralized architectures

Several architectures take advantage of filtering methods presented previously [18, 10, 7, 6]. Commercial platforms such as Google Earth [1] and Second Life [2] have recently known a tremendous craze from the public. The first allows navigation into and over a well-detailed model of the earth, with terrain and buildings. In doing this, it takes advantage of an adaptive streaming of terrain textures [20], associated with a multi-resolution wavelet-based representation for terrain, and static levels of detail for buildings. The second provides an advanced social-network service combined with general aspects of a metaverse. The Second Life client integrates modeling tools that allow users to create new components of the virtual environment.

In addition to the above, more than fifty online virtual worlds have appeared over the last few years, and are usually based on centralized architectures. Huge environments are generally partitioned, according to a grid, in regions that are managed by dedicated servers, called region servers. Unfortunately, in order to synchronize the game states of the world for all connected clients, most processes such as collision detection, physics computation, and animation, are executed on the server side. Moreover, the region server is the only source that can provide clients with the 3D models of the scene for its visualization. Thus, the number of clients that can navigate into a region managed by only one server is clearly limited.

3.3 Peer to Peer Architectures for Virtual Worlds

A centralized architecture cannot lead to a truly self-scalable solution, even with the use of multiple servers. Indeed, client-server architectures lead to prohibitive deployment and maintenance costs when it comes to very large scale applications with thousands of connected clients. On the other hand, thanks to their *self-adaptation* features, P2P network overlays have clearly proved to be an effective alternative to powerful servers.

Based on the fact that if peers have nearly the same viewpoint, they are likely to need the same data, geometric proximity is obviously the main criterion for setting up peer connectivity within virtual environments. However, finding and maintaining the appropriate peer connectivity is a very difficult problem in a dynamic environment and in the presence of churn, that is where peer viewpoints are allowed to move freely and when peers can disconnect or appear at any time. Solipsis [14, 15] and VON [12] are the first P2P layers providing a solution for 2D environments. In these P2P architectures, peers are connected to each other according to their current 2D position. Dedicated algorithms are used to achieve the global stability of the P2P network while fulfilling a global connectivity constraint (i.e. there must exist at least one path between each pair of peers).

Maintaining real-time peer connectivity in a n-dimensionnal space is much more complex. For this reason, Douglas et al. [5] have proposed a region-based approach using a distributed spatial data index in a multidimensional space to find nearby objects with which direct connections can be established. Finally, Cavagna et al. [4] show that server-less P2P networks can efficiently deal with very large environments, first by using well-suited descriptors to specify areas of interest for continuous levels of detail (for on-demand streaming), and second by having peers exchange information about their own serving capabilities (for self-regulating peer-upload bandwidth).

4 Decentralized Virtual Worlds Management

Our response to the demand for an efficient metaverse platform capable of 3D interactions among entities is the new version of Solipsis. Its key characteristic is the ability to distribute the cost associated with the management of its metaverse among the hosts participating in it. This is made possible through the decentralization of heavy processes like collision detection, computation of physics, as well as communication among the large number of nodes that participate in the metaverse. In the following, we present the main characteristics of the architecture and of the protocol enabling such decentralization.

4.1 Definitions

The Solipsis metaverse consists of a set of *entities*, each belonging to one of three categories: *avatars*, *objects*, and *sites*. Avatars are the main actors of the metaverse as they are the only entities that are capable of autonomous movement. In most cases they are virtual representations of the users of the metaverse and are directly controlled by them using the navigator platform described in Section 5. Objects, on the other hand, are virtual representation of entities from the real world such as furniture, books and whatever object may be moved or picked up by an avatar. Avatar and objects may also be robotic-like entities controlled by user-defined software components. Finally, sites constitute the basic building blocks of the metaverse and represents portions of the virtual space that may be occupied by objects or in which avatars can roam.

Avatars, objects, and sites are all associated with 3D-descriptions, each consisting of a mesh- or prims-based model, a set of textures, and, in the case of avatars or objects, also of an animation. Such 3D-descriptions constitute the basis for the rendering of entities in the navigator platform.

The state of an entity in the metaverse is determined by an *entity descriptor*, from now on referred to simply as *descriptor*. An entity's descriptor can contain information regarding its position,

UID	universal identifier of the entity
seqNum	sequence number
owner	identifier of the node managing the entity
type	site, avatar or object
loc	location in the 3D space
ori	orientation in the 3D space
shape	shape from a predefined set
box	bounding box of the object
R_p	perceptibility radius: distance from which the object is visible in the absence of obstacles
R_b	radius of smallest sphere enclosing the entity
objs_a	list of entities attached to the current one
f_1	first file of 3d-description
v_1	version number for first file
c_1	list of hosts that have cached v_1 of f_1
...	...
f_n	n-th file of 3d-description
v_n	version number for n-th file
c_n	list of hosts that have cached v_n of f_n
...	additional fields for progressive levels of details

Table 1: Example of a simple descriptor

its physical properties, and whatever is needed to display the entity and compute its interaction with the rest of the metaverse. For example, a descriptor containing a key frame can be used to synchronize animations, videos, and so on. Moreover, filtering descriptors defining areas (concentric spheres, hierarchical partitioning, cells) associated with a level of detail or a set of visible objects may be used to achieve efficient on-demand streaming while satisfying view-dependency criteria. Finally, descriptors may contain load-balancing information regarding, for example, the available resources or serving capabilities of a host [4]. Table 1 shows a sample descriptor containing the most relevant information. Although the descriptor is shown as a single object, the Solipsis implementation may split it into several descriptors regarding, for example, physical properties, location, or level of details in order to reduce communication cost.

4.2 Solipsis Hosts and Nodes

From a practical viewpoint, the Solipsis platform is distributed over a set of hosts that maintain information about every entity that is currently present in the metaverse. Each host is associated with a single instance of the Solipsis platform, and throughout its lifetime, it may create new entities or destroy previously created ones according to the requests of the navigator component. Also, similar to an entity, each host is associated with a unique identifier (UID).

Because each host may be responsible for several entities at the same time, we define a *(Solipsis) node* as the set of resources dedicated to the management of a given entity. Each node encapsulates information about the corresponding entity's descriptors as well as the threads of control responsible for managing all the interactions of the node with the rest of the Solipsis metaverse as well as with the navigator platform.

4.3 P2P Architecture

A fundamental aspect of Solipsis is its completely decentralized architecture designed to accommodate large numbers of entities, accessed by large numbers of users distributed over the Internet. The core of this decentralized architecture is a peer-to-peer overlay network, which is essentially a graph where nodes are connected by virtual logical links, each of which may consist of several physical links in the underlying IP network. These logical links may be laid out according to some proximity metric so as to enable efficient storage and retrieval of information.

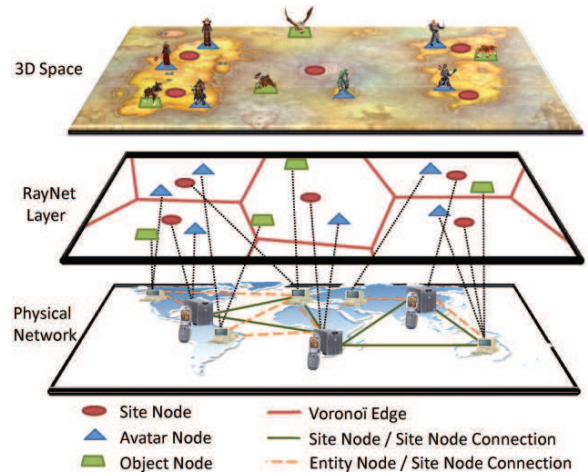

Figure 1: The p2p architecture showing the mapping of the Raynet layer on the virtual world layer according to the proximity of nodes in the virtual space. The host layer shows clients connected into a peer to peer scheme. Note that while the positions of avatars and objects are also shown, for clarity, on the Raynet layer, only site nodes actively join the RayNet and form the corresponding Voronoi diagram.

Recent years have seen the emergence of a large number of peer-to-peer overlay networks, with different structures and capabilities. In the context of Solipsis, we leverage the potentialities of peer-to-peer overlays by building our metaverse on top of RayNet [3]. Raynet is a multi-dimensional overlay network based on the concept of Voronoi tessellation. As shown in Figure 1, each node in the overlay is associated with a position in a multi-dimensional space. Neighborhood relationships between peers are then determined by the distance between the corresponding points in this space. Specifically, two Raynet nodes are neighbors in the overlay if the two corresponding points are neighbors in the Delaunay graph comprising all the points associated with Raynet nodes.

Given a set of generator points$\{p \in \Re^d\}$, the Delaunay graph is obtained as the adjacency graph of the corresponding Voronoi diagram, which in turn is a tessellation of the d-dimensional space $\Re^d$. Specifically, the cell in the tessellation associated with a point p_x is such that it contains all the points that are closer to p_x than to any other generator point in the set. This property enables Raynet to route to any node in the overlay by means of a simple greedy approach. Moreover, the addition of Kleinberg-like [16] long-range links allows this routing process to converge to its destination node in a polylogarithmic number of hops on average.

Within the context of Solipsis, we use the Raynet structure to organize *site nodes* into a 3-dimensional overlay in which the positions of nodes mimic those of the corresponding sites in the metaverse.[1] This allows our architecture to exploit very simple protocols to manage the interaction between avatars, objects and the sites they are currently located in as we describe in the following.

4.4 Decentralized physics computation

The choice of a peer-to-peer overlay such as Raynet is motivated by our goal to scale to metaverses containing very large numbers of entities, possibly gathered within the same site or within an otherwise small region of space. Organizing sites nodes into such an overlay, however, is only half of the picture. Solipsis also incorporates a fully decentralized protocol that distributes the cost associated with heavy computations such as those regarding collision

[1]Extensions of the metaverse architecture to higher dimensions are naturally supported by the Raynet overlay.

detection, physics, or animation.

Each avatar node is responsible for computing its own position based on physical criteria like its mass, momentum, and forces applied by the entities in its surroundings. Our decentralized approach allows it to achieve this result by taking into account only a small set of 3D models: those associated with the current site and with the entities that are in the avatar's immediate surroundings. For example, in the case of collisions, each avatar may immediately rule out the entities that are at a distance that is greater than the radii, R_b, of the spheres enclosing its and their own bounding boxes plus a configurable safety distance.

The basis for this decentralized computation of physics is a communication protocol that allows nodes to exchange information about entities' positions with three levels of heartbeat messages. Critical information that is required for the computation of collision detection is exchanged directly in a peer-to-peer fashion between the avatars that may potentially collide with each other, based on the size of their bounding boxes. Less critical information that is nonetheless necessary to provide the navigator with a clear snapshot of the current scene state is instead propagated indirectly, in a multi-hop fashion. Finally, information about distant objects or objects that have just joined the metaverse is propagated by site nodes to the avatars within their cells at a significantly lower frequency.

An avatar joining Solipsis contacts the site node responsible for its joining - or latest - location. Although the avatar is not part of the overlay, it can easily do this by using the Raynet to route a message containing its descriptor to the site node that is closest to its own position. This node then reacts by providing the avatar with the descriptors of the entities that are potentially visible from its location. From this point on, the avatar and site nodes exchange this information periodically to implement the low frequency updates described above. An analogous mechanism is used to manage the positions of object nodes. The use of the Raynet always allows avatars and objects to contact the right node as they move from site to site.

In addition, avatar and object nodes interact with the avatars and objects around them in order to exchange the critical up-to-date information about the entities that may collide with them. Also, while sending an update to a nearby node, avatar and object nodes also include information received from other nearby nodes that are not within collision distance of the destination node; thus implementing the second level of peer-to-peer updates.

4.5 Dynamic Object Management

While avatars are the main actors in the Solipsis environment, objects also play an important part as they can interact with avatars, be picked up, be moved from one location to another, or even move freely through space subject to gravity or other forces, or according to a script of animation. This requires our protocol to determine the current position of objects in addition to that of avatars. This is achieved through a combination of three mechanisms that depend on the state the object is currently in.

Let us consider an object that is currently stationary within a site S's cell. The position of the object is maintained by S's site node, which also maintains the main copy of the object's descriptor. As soon as an avatar tries to interact with the object, by moving it or picking it up, its avatar node sends a request to the site node to take over responsibility for the object. The site node grants such responsibility to the first avatar requesting it, according to the order in which requests reach the site node.

When permission to take over the object is granted, the object is dynamically detached from the site and attached to the avatar. The corresponding descriptors are updated, and the avatar nodes starts computing physics and maintaining the descriptor for the object. Note that physics computation need not be performed by site nodes, in that an object can only be attached to a site when it is in static equilibrium.

After interacting with the object, the avatar may pass the object on to another avatar, or leave it in the current or in a different site. In the former case, the recipient of the object takes over its management, and starts computing its physics, and animation, and updating its descriptor. In the latter, on the other hand, the object may be left in a stationary or in a dynamic state. If the object is stationary, then the site may take control of its descriptor and manage it without computing any physics. If, on the other hand, the object is moving, or subject to forces that are not in equilibrium, then a new object node with a corresponding thread is instantiated on the host associated with the avatar to which the object was attached. This new object node takes responsibility for managing the object's movements and updating its descriptor until it is picked up by a new avatar or until it reaches an equilibrium state at some site. It should be noted that a host may keep managing an object, even if the associated avatar is at a distant location. This allows hosts to manage robotic-like entities that are controlled by user-defined software components, e.g. by associating streaming video to objects.

4.6 Decentralized 3D-Model Sharing

The architecture we have described so far assumes that nodes are able to retrieve the 3D-description of entities in order to compute collisions and display the entities after filtering them based on the information in their descriptors. In the following, we describe our mechanism for maintaining and retrieving these 3D-descriptions. As with the rest of the protocol, our goal is to distribute the cost of managing 3D-descriptions among Solipsis participants.

3D-descriptions are managed by the nodes responsible for the corresponding entities, and may be downloaded by any node that needs to display the object or perform collision detection. However, having all hosts download a given 3D-description from the node responsible for its entity would place an unnecessary load on the corresponding host. We address this issue, by having Solipsis nodes cache previously downloaded descriptions. This allows them to offer them for download to other Solipsis nodes running on the same or remote hosts.

All the nodes residing on a host, store their own 3D-descriptions as well as those of other visible objects in a shared repository (depicted in Figure 2) that is accessible by all the nodes running on the same host. Moreover, each node records in its entity's descriptor a list of the hosts that currently hold a copy of any of the files that constitute the associated 3D description. Whenever a node caches a 3D-description, it informs the node responsible for the entity, which then updates the corresponding descriptor to include a reference to the new copy and then propagates it with its subsequent heartbeat messages. Similarly, when a node detects that a host listed in the descriptor does not have the current copy of an entity's description, it sends a message to the corresponding node, which updates the descriptor accordingly.

4.7 Managing Disconnections

The architecture of Solipsis is designed to tolerate unexpected disconnections of hosts throughout its operation. First, site nodes always cache the 3D-descriptions, in addition to the descriptors, of neighboring site nodes. This allows the RayNet to manage the disconnection of a site node by automatically assigning its management to any of the neighboring site nodes. The disconnection of an avatar or object node, on the other hand, is treated by attaching the object or avatar to its current site and by leaving it in a stationary state until the corresponding user reconnects.

5 The Navigator

A navigator must be able to create a huge metaverse and democratize its use. For this reason, we base our navigator on the Ogre

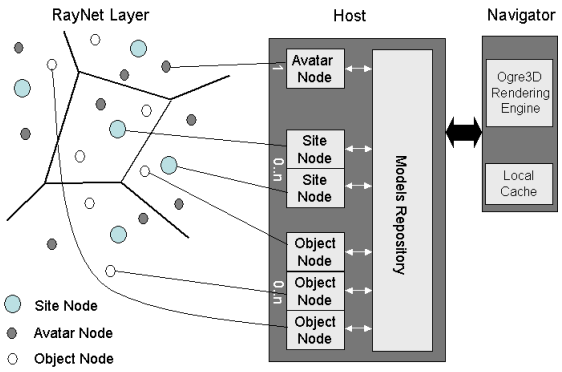

Figure 2: Nodes hosted on a proxy server to allow accesses to the metaverse on terminals with low resources such as mobiles.

Figure 3: Web based navigator embedding modeling tools, and providing a mapping of web pages as interactive textures.

3D rendering engine. Its robustness, efficiency, upgradability and openness are, in fact, essential features for a durable system. Moreover, Ogre 3D can be easily embedded in web pages thanks to a Mozilla or ActiveX plugin. Conversely, web pages can be mapped on 3D models as interactive textures thanks to the Navi library based on Gecko. Thus, the navigator allows a Web 2.0 navigation inside the Web 3D, as well as a Web 3D navigation inside the Web 2.0. Our view is, in fact, that Web 3D and Web 2.0 should integrate each other (see figure 3). Since the vitual universe belongs to all users, modeling tools are embedded within the navigator in order to create, modify or delete contents. At the moment, we focus on intuitive tools, but procedural, declarative, parametric and sketch-based modeling tools are obviously considered. Finally, to provide access to the virtual universe on mobile devices, nodes can be hosted by proxy servers, to reduce the amount of computation that has to be performed on terminals with scarce resources. Thus, hosts can compute collision detection, physics animation, data-exchange management, and viewpoint-based filtering, and then communicate with the navigator that only needs to render the virtual environment and interact with it (see figure 2).

6 Conclusions and perspectives

We presented our vision for a decentralized architecture for virtual environments. Our solution is based on an n-dimensional Voronoi-based peer-to-peer network, called RayNet. This allows it to distribute communication and computational cost among the various nodes present in the virtual space. Our architecture enables the decentralization of complex tasks related to physic-realistic modeling. To achieve this, nodes preliminarily filter the set of avatars and objects that may collide with their own and then evaluate collisions and physics for a small set of entities. Our solution also supports dynamic objects that may be picked up and dropped by avatars or controlled by means of user-defined software. Moreover, it enables adaptive streaming of 3D models in a completely decentralized fashion while adapting streamed data to avatars' viewpoints. Also, the use of an n-dimensional overlay allows us to extend the metaverse to support social or semantic proximity between object or avatar nodes. Access to the metaverse is made possible by a navigator that may run as a stand-alone platform or embedded within a web page. The navigator exploits interactive texturing to enable the visualization of Web 2.0 components in the virtual world and it integrates tools for modeling new 3D contents. Moreover, it supports operation on resource-scarce devices such as mobile phones. Our project team is currently working on a fully open-source implementation of the Solipsis architecture, and we hope that the community of users will help us improve Solipsis and enable us to evaluate its effectiveness in a world-wide testbed.

Acknowledgements

First, the authors wish to thank all people who contributes to the Solipsis project as well as all pioneers of the decentralization of virtual environments whithout who this project would not be also advanced: M. Keller, M. Simon, M. Cavagna, M. Bouville and M. Beaumont. This work was supported in part by a French collaborative R&D project (ANR-RIAM) leaded by Orange Labs, funded by ANR and Media & Networks cluster of Brittany, involving IRISA, Rennes 2 University, Archivideo and Artefacto.

References

[1] http://earth.google.com/.
[2] http://www.secondlife.com/.
[3] O. Beaumont, A.-M. Kermarrec, and E. Rivire. Peer to peer multidimensional overlays: Approximating complex structures. In *OPODIS, 11th International conference on principles of distributed systems*, 2007.
[4] R. Cavagna, C. Bouville, and J. Royan. P2p network for very large virtual environment. In *VRST*, pages 269–276. ACM, 2006.
[5] S. Douglas, E. Tanin, and A. Harwood. Enabling massively multiplayer online gaming applications on a p2p architecture. In *Proceedings of the IEEE International Conference on Information and Automation*, pages 7–12. IEE, 2005.
[6] E. Frécon and M. Stenius. Dive - a scalable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal (Special issue on Distributed Virtual Environments)*, 5, 1998.
[7] T. A. Funkhouser. RING: A client-server system for multi-user virtual environments. In *Symposium on Interactive 3D Graphics*, pages 85–92, 209, 1995.
[8] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 247–254, New York, NY, USA, 1993. ACM.
[9] P. Gioia, O. Aubault, and C. Bouville. Real-time reconstruction of wavelet-encoded meshes for view-dependent transmission and visualization. *IEEE Trans. Circuits Syst. Video Techn.*, 14(7):1009–1020, 2004.
[10] C. Greenhalgh and S. Benford. MASSIVE: A distributed virtual reality system incorporating spatial trading. In *International Conference on Distributed Computing Systems*, pages 27–34, 1995.
[11] H. Hoppe. Progressive meshes. *Computer Graphics*, 30(Annual Conference Series):99–108, 1996.
[12] S.-Y. Hu and G.-M. Liao. Scalable peer-to-peer networked virtual environment. In *NetGames '04: Proceedings of 3rd ACM SIGCOMM*

workshop on Network and system support for games, pages 129–133. ACM, 2004.

[13] M. Isenburg and P. Alliez. Compressing polygon mesh geometry with parallelogram prediction. In *Visualization conference proceedings*, pages 141–146, 2002.

[14] J. Keller and G. Simon. Toward a peer-to-peer shared virtual reality. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 695–700. IEEE Computer Society, 2002.

[15] J. Keller and G. Simon. Solipsis: A massively multi-participant virtual world. In *Int. Conf. Parallel and Distributed Techniques and Applications*, pages 262–268, 2003.

[16] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[17] D. P. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3):24–35, 2001.

[18] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz. Npsnet: A network software architecture for large-scale virtual environment. *Presence*, 3(4):265–287, 1994.

[19] M. Pesce, P. Kennard, and A. Parisi. Cyberspace. In *First International Conference on WWW*, 1994.

[20] C. C. Tanner, C. J. Migdal, and M. T. Jones. The clipmap: a virtual mipmap. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158. ACM, 1998.

[21] S. J. Teller and C. H. Séquin. Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Comput. Graph.*, 25(4):61–70, 1991.

[22] P. Wonka, M. Wimmer, and F. Sillion. Instant visibility. In *EuroGraphics*. Eurographics, A. Chalmers and T.-M. Rhyne, 2001.