Massively Distributed Virtual Worlds
A Formal Approach

Patrik Fuhrer and Jacques Pasquier-Rocha

University of Fribourg
Department of Informatics
Rue P.-A. de Faucigny 2
CH-1700 Fribourg
Switzerland
patrik.fuhrer@unifr.ch
WWW home page: http://diuf.unifr.ch/∼fuhrer/

Abstract. The aim of this paper is to define a formal mathematical model for distributed virtual worlds. Starting from basic requirements, an abstract model is built. The latter attempts to be as general as possible in order to be generic and support different concrete instantiations. Two such concrete models are then presented. Finally, strategies for events distribution and some implementation considerations are discussed.

Keywords: Virtual World, Abstract Model, Distributed Events

Contents

1 Introduction

Since several years, our research group is involved with the task of designing and implementing *distributed object-oriented virtual worlds*. Our work lead to the implementation of MaDViWorld[1], an extensible software framework supporting the creation of such shared virtual environments. While previous papers presented MaDViWorld's software architecture and some implementation choices ([5], [4], [7], [6]), the present one defines and discusses the formal theoretical underlying model of our approach. In order to best achieve this goal, it is organized as follows. The introductory section is dedicated to a typical scenario, which should be possible in a virtual world. Building on this story, we then identify and extract the main concepts involved. Section 2 concentrates on the definition of the abstract model. Section 3 presents some possible instantiations of the model. Section 4 treats the issue of the interactions that occur between the components of the world. Finally, some implementation considerations conclude the paper.

1.1 The Vision

Suppose we have a virtual world, a user wants to "live" in. As the user strolls through the world, she discovers it and its components, objects and other users like her. Suddenly, she finds an interesting object, a fibonacci number calculator. She can use it to compute some numbers of the famous series. Then, she goes on with her world discovery. Doing so, she comes to a place where she sees two other users playing a "tic-tac-toe" game and a little crowd watching them. She joins the observers and, after a while, she says to her neighbor: "Do you want to play this game with me?" As the other agrees, they go to another place where the user has placed a copy of the previous board game and they start to play. When they are finished, she puts the "tic-tac-toe" object back in her bag and leaves the other user.

1.2 The Key Concepts

We consider that the main participants of our model emerging from this simple scenario are:

- **Avatar**: The human users need representations in the virtual world and are therefore personified by an avatars. Through her avatar a user can walk, "fly", look around the virtual world, manipulate objects and perform virtual actions. In other words, avatars allow users to interact with the virtual world and other users and also allow navigation through the world. We see one's avatar as being your representative in Cyberspace. In text-based virtual realities, such as MUDs, one's avatar consists of a short description which is displayed to other users who have their avatars 'look' at you. In a 3D graphical world, the avatar can take the shape of an animated cartoon or of your favorite fantasy hero.
- **Object**: In the discussed virtual world, there are objects, not just simple passive data objects, but active objects the avatar can execute (e.g. the fibonacci calculator). These objects can even be multi-user (e.g. the two players "tic-tac-toe" board) and can have many observers like in our little story. Last, but not least, objects can be copied and/or moved by the avatar.
- **Location**: Avatars and objects have a location. This concept is natural and necessary, since it supports navigation.
- **Navigation**: The action of going from a given location to another. Avatars can do that, either if there is a **link** between these two locations, or directly if they know the **address** of the subspace.

[1] MaDViWorld stands for Massively Distributed Virtual Worlds

- **Subspaces**: Each location can be seen as a subspace of the whole virtual world. It is natural to consider that the shared virtual space is composed of many different subspaces. Furthermore, the avatars and objects are always **contained** within one given subspace.
- **Event**: The avatar has to be aware of its environment. This awareness is achieved by the concept of events. An avatar moving, a played move in the "tic-tac-toe" game are simple examples of such events.
- **Event producer**: An event always has a source which produces it. For instance, the game could produce a "game finished event".
- **Event consumer**: An event can be catched and interpreted by an event consumer, which then reacts properly or simply ignores it. For instance, the player of the game understands that the game is finished.
- **Event propagation**: Each event has an event propagation space. This space is a delimited zone around an event producer, within which an event consumer will be aware that the event has occurred.

The following sentence, simply resumes the last four concepts. "An event is consumed by each interested event consumer in the event propagation space around the event producer."

2 The Abstract Model

All the concepts above need to be integrated in our theoretical model. Before formalizing them in the most general way, however, a brief "non-mathematical" summary should help the reader:

1. At the core of the model, there is the whole virtual space $\mathcal{V}$. $\mathcal{V}$ has a global metric space associated to it, which allows subspaces to have a location and a volume.
2. The subspaces are inhabited by avatars, objects and link extremities. The latter have a location and a volume relative to the local metric space associated with their container.
3. Further axioms and rules achieve the description of the relations between all these concepts.

2.1 The global virtual space

- Let $\mathcal{V} := (\mathcal{V}_V, \mathcal{V}_A, \mathcal{V}_N)$ be the whole virtual world. It represents the entire shared virtual space. It is seen as a *directed graph*[2] consisting of a set of vertices $\mathcal{V}_V$ and a set of arcs $\mathcal{V}_A$. The vertices are also called *nodes* or *points*; the arcs could be called *directed edges*. An arc is an ordered pair of vertices and is expressed by:

$$v_i \rightarrow v_k \text{ with } v_i, v_k \in \mathcal{V}_V \tag{1}$$

- For a given arc $v_i \rightarrow v_k$, v_i is called the *head* and v_k is called the *tail* of the arc. They are respectively defined by the following functions:

$$h(v_i \rightarrow v_k) := v_i \text{ and } t(v_i \rightarrow v_k) := v_k \tag{2}$$

- $\mathcal{V}_V$ is the set of all the *subspaces* v_i composing the world $\mathcal{V}$:

$$\mathcal{V}_V := \bigcup_{i \in I} \{v_i\} \text{ where } I \text{ is a set of indices} \tag{3}$$

[2] Directed graphs are well introduced in [1] and further discussed in [2].

- $\mathcal{V}_A$ is the set of all the *links* between the subspaces v_i of $\mathcal{V}_V$. It is of the form:

$$\mathcal{V}_A := \{v_i \rightarrow v_k : v_i, v_k \in \mathcal{V}_V\} \tag{4}$$

- $\mathcal{V}_N$ is the *namespace* used for the addresses of the rooms.
- Each subspace has an address given by the function:

$$\lambda : \mathcal{V}_V \longrightarrow \mathcal{V}_N \tag{5}$$

- Each subspace v_i of $\mathcal{V}_V$ has a *location* in $\mathcal{V}$ given by the function:

$$\overline{\pi} : \mathcal{V}_V \longrightarrow (\overline{\Sigma}, d_{\mathcal{V}}) \tag{6}$$

- Each subspace v_i of $\mathcal{V}_V$ has a *volume* in $\mathcal{V}$ given by the function:

$$\overline{\rho} : \mathcal{V}_V \longrightarrow \mathcal{P}((\overline{\Sigma}, d_{\mathcal{V}})) \tag{7}$$

  Where $\mathcal{P}(\overline{\Sigma})$ is the powerset (set of all subsets) of $\overline{\Sigma}$.
- $(\overline{\Sigma}, d_{\mathcal{V}})$ is a metric space[3]. So there must be a metric $d_{\mathcal{V}}$ defined on the location space $\overline{\Sigma}$ of $\mathcal{V}$.

$$d_{\mathcal{V}} : \overline{\Sigma} \times \overline{\Sigma} \longrightarrow \mathbb{R} \tag{8}$$

2.1.1 Avatars, objects and transport points

- Let A be the set of all *avatars* α_i living in the subspaces of $\mathcal{V}_V$:

$$A := \{\alpha_1, \alpha_2, ..., \alpha_n\} \tag{9}$$

- Let Ω be the set of all *objects* ω_i populating the subspaces of $\mathcal{V}_V$:

$$\Omega := \{\omega_1, \omega_2, ..., \omega_m\} \tag{10}$$

- Let us associate to each link $x \in \mathcal{V}_A$ the notion of a *leaving point* and an *entry point*:

$$\begin{aligned} x_{e^{\rightarrow}} &\text{ is the leaving point of } x \\ x_{e^{\leftarrow}} &\text{ is the entry point of } x \end{aligned} \tag{11}$$

- Let the set of all leaving and entry points be regrouped in the set of *transport points* E defined by:

$$E := \bigcup_{x \in \mathcal{V}_A} \{x_{e^{\leftarrow}}, x_{e^{\rightarrow}}\} := \{\varepsilon_1, ..., \varepsilon_n\} \tag{12}$$

  The cardinality, i.e. the number of elements, of E, noted $\#E$, is given by the following relation: $\#E = 2 \cdot \#\mathcal{V}_A$
- Let Θ be the set of all entities living in the virtual world, including avatars, objects and transport points:

$$\begin{aligned} \Theta &:= A \cup \Omega \cup E = \{\alpha_1, ..., \alpha_n, \omega_1, ..., \omega_m, \varepsilon_1, ..., \varepsilon_{\#E}\} \\ &=: \{\theta_1, \theta_2, ..., \theta_{n+m+\#E}\} \end{aligned} \tag{13}$$

- The avatars, objects and transport points must be located in *exactly one subspace* of the world. This means, that each element of Θ has a location, i.e. its container in $\mathcal{V}$ given by the function π:

$$\pi : \Theta \longrightarrow \mathcal{V}_V, \text{ so that } \theta \in \pi(\theta) \in \mathcal{V}_V \tag{14}$$

[3] For a general introduction about metric spaces cf. [3] or [8].

- Naturally the following statement is always verified:

$$\begin{aligned} \forall \theta_j \in A \cup \Omega \; \exists! v_i \in \mathcal{V}_V \text{ with } \pi(\theta_j) = v_i \\ 1 \leqslant j \leqslant m+n \text{ and } i \in I \end{aligned} \tag{15}$$

- Furthermore, the following property must be respected:

$$\begin{aligned} \forall x \in \mathcal{V}_A \quad \pi(x_{e\rightarrow}) = h(x) \\ \text{and } \pi(x_{e\leftarrow}) = t(x) \end{aligned} \tag{16}$$

2.1.2 The local subspaces

- Let A_{v_i} be the set of all avatars living in a given subspace v_i of $\mathcal{V}_V$:

$$A_{v_i} := \{x \in A : \pi(x) = v_i\} \tag{17}$$

- Let Ω_{v_i} be the set of all objects populating a given subspace v_i of $\mathcal{V}_V$:

$$\Omega_{v_i} := \{x \in \Omega : \pi(x) = v_i\} \tag{18}$$

- Let E_{v_i} be the set of all transport points located in a given subspace v_i of $\mathcal{V}_V$:

$$E_{v_i} := \{x \in E : \pi(x) = v_i\} \tag{19}$$

- Let Θ_{v_i} be the set of all objects located in v_i:

$$\Theta_{v_i} := A_{v_i} \cup \Omega_{v_i} \cup E_{v_i} := \{x \in \Theta : \pi(x) = v_i\} \tag{20}$$

- Each element of Θ_{v_i} has a *location relative* to its container v_i, given by the function:

$$\pi_{v_i} : \Theta_{v_i} \longrightarrow (\Sigma_{v_i}, d_{v_i}) \tag{21}$$

- Each element of Θ_{v_i} has a *volume relative* to its container v_i, given by the function:

$$\rho_{v_i} : \Theta_{v_i} \longrightarrow \mathcal{P}((\Sigma_{v_i}, d_{v_i})) \tag{22}$$

- (Σ_i, d_{v_i}) are metric spaces. So there must be a metric d_{v_i} defined on each location space Σ_{v_i} of v_i.

$$d_{v_i} : \Sigma_{v_i} \times \Sigma_{v_i} \longrightarrow \mathbb{R} \tag{23}$$

- The elements of $\mathcal{V}_V$, i.e the subspaces, as well as those of $\Omega \cup A$, i.e. the objects and avatars, can fire events. Each event source s has an event propagation space defined by the function p:

$$\begin{aligned} p(s,k) := \{x \in \varsigma :=& \pi(s) : d_\varsigma(\pi_\varsigma(s), \pi_\varsigma(x)) \leqslant k\} \\ & \text{with } k > 0 \in \mathbb{R} \text{ constant} \end{aligned} \tag{24}$$

  In Subsection 4 the events are further explained.

2.2 Further Remarks

- Within a running model, the virtual world $\mathcal{V}$ may avoid overlapping and collision of its subspaces v_i. In this case the following invariant is always verified:

$$\overline{\rho}(v_k) \cap \overline{\rho}(v_l) = \emptyset, \quad v_k, v_l \in \mathcal{V} \text{ and } \forall k \neq l \tag{25}$$

- Each subspace v_i has its own location function and coordinates system.
- Within a running model, a subspace v_i may avoid overlapping or collision of the objects θ_k it contains. In this case the following invariant is always verified:

$$\rho_{v_i}(\theta_k) \cap \rho_{v_i}(\theta_l) = \emptyset, \quad \theta_k, \theta_l \in \Theta_{v_i} \text{ and } \forall k \neq l \tag{26}$$

- Within a running model, only symmetric links could be allowed. Following implication should then be always verified:

$$v_i \rightarrow v_k \in \mathcal{V}_A \Longrightarrow v_k \rightarrow v_i \in \mathcal{V}_A \tag{27}$$

  In this case, let us define the notion of symmetric arc and note $v_i \leftrightarrow v_k \in \mathcal{V}_A$.
- If a running model satisfies (27), then it could also always guarantee:

$$\pi_{v_i}((v_i \rightarrow v_k)_{e\rightarrow}) = \pi_{v_i}((v_k \rightarrow v_i)_{e\leftarrow}) \tag{28}$$

2.3 Remarks about the time

- The virtual world $\mathcal{V}$ is in constant evolution through the time. Further, we consider that the time is discrete. So all sets and functions defined above depend actually on a parameter t defining the instant t. Thus, we should have in fact $\mathcal{V}_{At}$, $\mathcal{V}_{Vt}$, I_t, A_t, Ω_t, ε_t, Θ_t, $\overline{\pi_t}$, $\overline{\rho_t}$, π_t, $\theta_{v_i t}$, $\pi_{v_i t}$, $\rho_{v_i t}$ and $p_t(s,k)$.
- Each event occurs at an instant, so there are no simultaneous events in the model.
- Each event corresponds to a transition between two instants $t \longrightarrow t+1$. An event is considered either at the local level, or at the global level or both.
- In order to consider events at the global level, a *global clock* T is needed.
- The local level events can either use the global clock if there is one or define their own *local clock* τ.
- A non exhaustive list of possible *global level* transitions (i.e. events) would be:
  - the creation of a new subspace
  - the destruction of a subspace
  - an avatar who moved from one subspace into another
  - a new avatar who joins the world (logs in)
  - an avatar who leaves the world (disconnects)
  - a object added to a subspace
  - the creation of a link between two subspaces
  - the destruction of a link between two subspaces

  A non exhaustive list of possible *local level* transitions (i.e. events) would be:
  - an avatar who arrives in the subspace
  - an avatar who leaves the subspace
  - an avatar who says something to another avatar in the room
  - an object added to the subspace
  - an internal state change of an object in the subspace
  - the creation of a link between the subspace and another

2.4 Final definition

To summarize, one can define a virtual world system VW as a 5-tuple:

$$VW = (\mathcal{V}, \overline{\rho}, \Theta, \overline{\pi}, (\overline{\Sigma}, d_{\mathcal{V}})) \tag{29}$$

And each subspace v_i, element of $\mathcal{V}_V$, can be defined as a 4-tuple:

$$v_i = (\Theta_{v_i}, \rho_{v_i}, \pi_{v_i}, (\Sigma_{v_i}, d_{v_i})) \tag{30}$$

For sake of simplicity, we consider the whole model at a fixed time t and omit the time parameter.

3 Model instantiation

In order to instantiate this abstract model, we have to define the functions $\overline{\pi}$,π_{v_i},$\overline{\rho}$,ρ_{v_i}, the location spaces $\overline{\Sigma}$, Σ_{v_i} and the respective metrics $d_{\mathcal{V}}$,d_{v_i}. The parameter k of formula (24) has to be fixed as well.

3.1 The "natural" instantiation of the model

The "natural" way to instantiate the model, consists in considering an arbitrary large number of subspaces located in the three-dimensional space.

$$\mathcal{V}_V := \bigcup_{i \in I} \{v_i\} \subseteq \mathbb{R}^3$$

Here we define $\overline{\Sigma} = \mathbb{R}^3$, and the function $\overline{\pi}$ gives the position of the subspaces v_i in the natural coordinates. The function $\overline{\rho}$ associates each subspace v_i to the 3D volume it occupies in the space. These functions are illustrated on Figure 1.
The functions

$$\pi_{v_i} : \Theta_{v_i} \longrightarrow \Sigma_{v_i} := (\mathbb{R}^3, d) \ \forall i \in I$$

associate each element of Θ_{v_i} to its position in natural coordinates. The functions ρ_{v_i} associate each element of Θ_{v_i} to a 3D volume it occupies in the subspace. We then define the traditional euclidian metric d on all location spaces $\Sigma_{v_i} = \overline{\Sigma} = \mathbb{R}^3$:

$$d_{\mathcal{V}}(x, y) = d_{v_i}(x, y) = d(x, y) = \sqrt{\sum_{j=1}^{3} (x_j - y_j)^2}$$
$$\forall i \in I, \forall x := (x_1, x_2, x_3), y := (y_1, y_2, y_3) \in \mathbb{R}^3$$

The event propagation space is defined by the function:

$$p(s, k) := \{x \in \pi(s) : d(\pi_{\pi(s)}(x), \pi_{\pi(s)}(s)) \leqslant k\}$$
$$\text{with } k > 0 \in \mathbb{R} \text{ constant}$$

Figure 1 illustrates a concrete example. On this figure one can see the following relations:

$$\mathcal{V}_V := \bigcup_{i=1}^{8} \{v_i\} \subseteq \mathbb{R}^3$$

$$\mathcal{V}_A = \{v_3 \leftrightarrow v_5, v_5 \leftrightarrow v_6\}$$
$$\pi(\omega_1) = \pi(\alpha_3) = v_6$$

The observations below can be made:

- $p(\omega_1)$ is delimited by the dotted line.

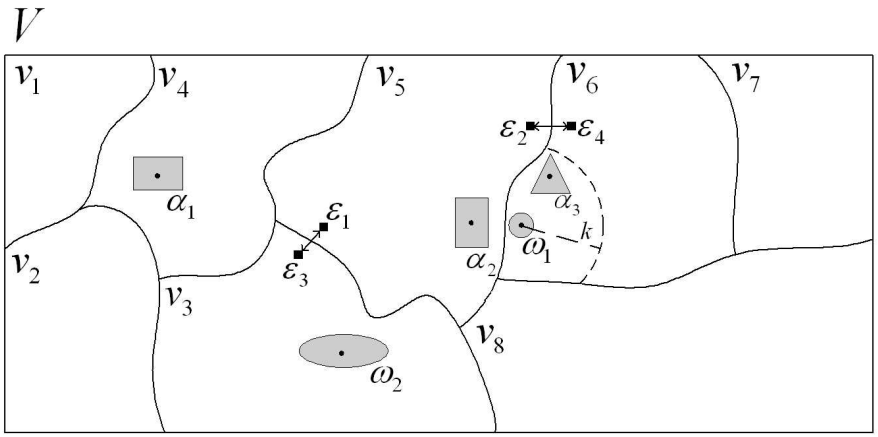

Figure 1: The "natural" instantiation of the model.

- $\rho(\alpha_1)$ is represented by the shadowed zone around $\pi_{\pi(\alpha_1)}(\alpha_1) = \pi_{v_4}(\alpha_1)$, which is represented by a black dot.
- The entry points are represented by a black square (cf. $\varepsilon_1 := \pi_{v_5}((v_3 \rightarrow v_5)_{e\leftarrow})$).
- The world shown in this example respect both properties (25) and (26).
- Furthermore, properties (27) and (28) are also respected and only links between two adjacent subspaces are allowed. They support the metaphor of doors.

3.2 The MaDViWorld instantiation of the model

The implemented MaDViWorld framework supports a more pragmatic instantiation of the model. Indeed, one considers only a finite number of subspaces, which have no "real" location.

$$\mathcal{V}_V = \bigcup_{i=1}^{l} \{v_i\}$$

We define $\overline{\Sigma} := \{1\}$, and the function $\overline{\pi}$ gives the unique and same position for all subspaces v_i, i.e. 1. We define the trivial metric $d_{\mathcal{V}}$ on the location space $\overline{\Sigma} := \{1\}$:

$$d_{\mathcal{V}} : \{1\} \times \{1\} \longrightarrow \mathbb{R}, d(1,1) = 0$$

The function $\overline{\rho} \equiv \overline{\pi}$. This means that the different subspaces composing the virtual world have no position and occupy no space.
For all subspaces $v_i, 1 \leqslant i \leqslant l$, we define the same degenerated location functions π_{v_i} :

$$\pi_{v_i} : \Theta_{v_i} \longrightarrow (\Sigma_{v_i}, d) \text{ where } \Sigma_{v_i} := \{1,2,3\}$$

$$\pi_{v_i}(x) := \begin{cases} 1 & \text{if } x \in A_{v_i} \ (x \text{ is an avatar}) \\ 2 & \text{if } x \in \Omega_{v_i} (x \text{ is an object}) \\ 3 & \text{if } x \in \varepsilon_{v_i} (x \text{ is a transport point}) \end{cases}$$

We define the same simple metric d on all location spaces $\Sigma_{v_i} = \{1,2,3\}, \forall i \quad 1 \leqslant i \leqslant l$:

$$d_{v_i} = d : \{1,2,3\} \times \{1,2,3\} \longrightarrow \mathbb{R},$$

$$d(x,y) := \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}, \quad \forall i, 1 \leqslant i \leqslant l$$

The function $\rho_{v_i} \equiv \pi_{v_i}$. This means that the elements in the subspace have no "real" position and no volume. We can just say that there are three different lists, one of links, one of objects and one of avatars.
The event propagation space is defined by the function:

$$p(s) := \pi(s)$$

This means that an event fired by an object s can be consumed by any other object in the same subspace.

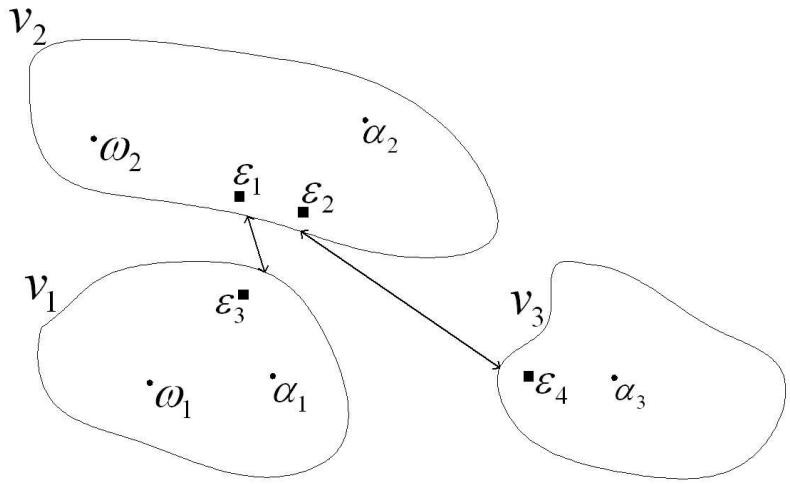

Figure 2: The MaDViWorld instantiation of the model.

Figure 2 sketches an example of a concrete world within this model. On this figure one can see the following relations:

$$\mathcal{V}_V = \bigcup_{i=1}^{3} \{v_i\}$$

$$\mathcal{V}_A = \{v_2 \leftrightarrow v_1, v_2 \leftrightarrow v_3\}$$

$$\pi(\omega_1) = \pi(\alpha_1) = v_1$$

$$p(\omega_1) = \pi(\omega_1) = v_1$$

$$\rho_{v_1}(\omega_1) = \pi_{v_1}(\omega_1)$$

The two observations below can be made:

- Properties (27) and (28) are respected and links support the metaphor of "teleportation gates".
- As the location function in this model is partially degenerated (all avatars share the same location, all objects share the same location,...), it is clear that neither property (25) nor (26) is respected.

4 Events And Interaction

In [7] we hinted how the event traffic is achieved inside a virtual world. It is worth clarifying these ideas by formalizing them and by giving some definitions.

4.1 Global view

The four main concepts supporting interaction between the components of a virtual world are explained below:

- The *event source* or *event producer* is the object that fires an event. It can be an avatar, an object or a subspace.
- The *event listener* works on behalf of a given event consumer. Its role is to "catch" events that occur in the event consumer's environment.
- The *event consumer* is the object that is interested in events and that will consume them. It reacts to received events. If an object c is interested in events fired by a given source object s, it has to register an event listener l to s. An event consumer can have several event listeners working for it.
- The *event* is an object describing an action or a state change.

4.2 Formalization

Let us write l^c for a listener l working on behalf of the event consumer c.
For a given event source s, we define the set of associated registered event listeners s_l as follows:

$$s_l = \{l_1^{c_1}, l_2^{c_1}, ..., l_n^{c_m}\} \text{ where } c_i \in p(s) \quad \forall i, 1 \leqslant i \leqslant m$$

An event $\varepsilon_{s,\Delta}^{i,t}$ has several attributes:

- A sequence number i (i^{th} event produced by the source s)
- A timestamp t giving the time at which the event occured.
- A set of recipients $\Delta \subseteq s_l$
- A source s
- A type $\tau(\varepsilon)$ which can be a (avatar), s (subspace) or o (object).

As expected, avatars can fire events of type a, objects can fire events of type o and subspaces can fire events of type s. Furthermore, subspaces can forward received events of type a or o.
The listeners can be registered to an event producer by the event consumers according to one of the following policies γ:

1. γ_{all} : "Inform me of all events."
2. γ_{me} : "Inform me only of events explicitly addressed to me."
3. γ_a : "Inform me only of event of type a."
4. γ_o : "Inform me only of event of type o."
5. γ_s : "Inform me only of event of type s."

Therefore the set s_l can be decomposed as follows:

$$s_l = s_l^{\gamma_{\text{all}}} \cup s_l^{\gamma_{\text{me}}} \cup s_l^{\gamma_a} \cup s_l^{\gamma_o} \cup s_l^{\gamma_s}$$

where s_l^γ is the subset of listeners registered with the policy γ.

4.3 Benefits

Figure 3 shows an event source firing events to its registered event listeners. This model allows for the optimization of the event traffic between the different objects of the virtual world. Indeed, the event source does not fire events to listeners, which do not care about these events. Furthermore, the event listeners do not have to check if the events they received are relevant to them or not.

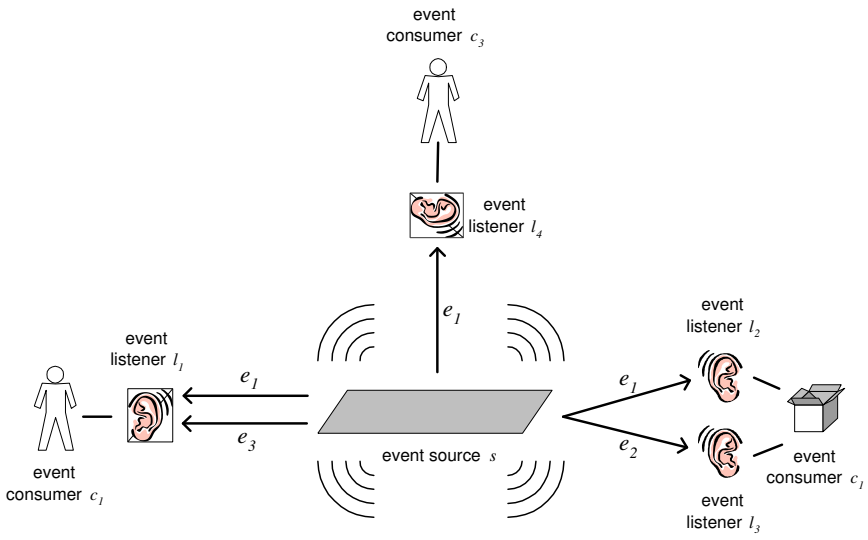

Figure 3: An event source and its listeners.

5 Implementation Considerations

The abstract conceptual model presented in Section **??** is general and it has been shown how to instantiate it in order to obtain a concrete model that can be implemented. Among the many possible instantiations of the model, two were presented in Subsections 3.1 and 3.2.

As already mentioned in the Introduction, the aim of our work is to implement *distributed* multi-user virtual worlds. Let us see if the model is suitable for distributed virtual worlds as well.

Pertaining to the time (cf. Subsection 2.3), if the world manages and diffuses events consistently at the global level, there must be a central clock. One, however, can restrict events to the subspace containing the event source. Thus, only independent local clocks are required and no centralized clock server is needed.

We identify properties (6) and (7) as the only ones presenting a real problem if there is no central server. Indeed, in order to implement these two functions, there must be a central unit which knows all the subspaces existing at every moment and their locations. This central unit maintains the coherence of the world relative to its chosen metric. That is the reason why, in MaDViWorld, we made the choice that the different subspaces composing the virtual world have no position and occupy no space. Loosing only these two restrictions, we developed a completely distributed virtual world in a quite natural manner.

Regarding the deployment, we chose not to spread a single subspace over several machines. Indeed, the model offers a sufficient fine grained granularity and there is no need to split a subspace in several parts. Thus in our implemented solution, one can have several rooms running on the same server, but a single room cannot run on several servers. This is shown in the next section.

Concerning the subspaces "internal behavior", the model presents no constraints to each of them to manage their objects and topology. The subspaces, which are the unit of decomposition of our whole virtual world, are completely free to manage their objects and topologies as they want. In our actual prototype, we also chose to have the simplest possible subspaces: the objects in them have no position and do not occupy a "physical" volume in the space. The fact that 3D features and the social aspects of interactions are not the first priority of the MaDViWorld project, justifies this choice. The model, however, allows for the addition

of more realistic or sophisticated topologies.

6 Future Work

Building on the formal model presented in Section 2, on its MaDViWorld's instantiation of Subsection 3.2, and the events propagation description of Section 4 , we shall next identify the major methods of MaDViWorld's main abstract classes, as well as specify formally their roles and signatures. A very first attempt at identifying these methods is summarized in Table 1 below. For sake of simplicity, detailed methods signatures have been omitted.

Avatar	
`getInformation();`	Returns the description of the avatar.
`getCurrentRoom();`	Returns the current room where the avatar is.
Object	
`getContainer();`	Returns the container of the object.
`setContainer();`	Sets the container of the object.
`getUI();`	Gets a graphical user interface to interact with the object.
Room	
`addObject();`	Sets a new object into the room.
`getObject();`	Returns a given object in the room.
`removeObject();`	Removes a given object from the room.
`getObjects();`	Returns a list of the objects in the current room.
`connect();`	Sets a given avatar into the room.
`disconnect();`	Removes a given avatar from this room.
`getAvatars();`	Returns a list of the avatars in the current room.
`addDoor();`	Adds a door to a given room.
`getDoors();`	Returns a list of doors to other rooms.
Event Producer	
`register();`	Registers an interested event consumer with this producer.
`unregister();`	Unregisters a registered event consumer with this producer.
Event Consumer	
`notify();`	Notifies the event consumer of an event.

Table 1: Some important method candidates of the main classes.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Computer Science and Information Processing. Addison-Wesley, 1983.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

[3] O. Forster. *Analysis 2: Differentialrechnung im* $\mathbb{R}^n$ *- Gewönliche Differentialgleichungen*. Vieweg Studium, 5th edition, 1984.

[4] P. Fuhrer, G. K. Mostéfaoui, and J. Pasquier-Rocha. The madviworld software framework for massively distributed virtual worlds: Concepts, examples and implementation solutions. *Department of Informatics Internal Working Paper no 01-23, University of Fribourg, Switzerland*, 2001.

[5] P. Fuhrer, G. K. Mostéfaoui, and J. Pasquier-Rocha. Madviworld: a software framework for massively distributed virtual worlds. *Software - Practice And Experience*, 32(7):645–668, June 2002.

[6] P. Fuhrer and J. Pasquier-Rocha. Massively distributed virtual worlds: a framework approach. *Department of Informatics Internal Working Paper no 02-16, University of Fribourg, Switzerland*, 2002.

[7] P. Fuhrer and J. Pasquier-Rocha. Massively distributed virtual worlds: A framework approach. In E. A. Nicolas Guelfi and G. Reggio, editors, *Scientific Engineering for Distributed Java Applications*, volume 2604 of *Lecture Notes in Computer Science*, pages 111–121. International Workshop, FIDJI 2002 Luxembourg-Kirchberg, Luxembourg, November 2002, Springer-Verlag, March 2003.

[8] W. Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, Tokio, 3 edition, 1976.