

An Abstraction for Awareness Management in Collaborative Virtual Environments

Miguel Antunes
INESC-ID/IST Technical
University of Lisbon
Rua Alves Redol n°9,
1000-029 Lisboa, PORTUGAL
Miguel.Antunes@inesc-
id.pt

António Rito Silva
INESC-ID/IST Technical
University of Lisbon
Rua Alves Redol n°9,
1000-029 Lisboa, PORTUGAL
Rito.Silva@inesc-id.pt

Jorge Martins
INESC-ID/IST Technical
University of Lisbon
Rua Alves Redol n°9,
1000-029 Lisboa, PORTUGAL
Jorge.B.Martins@inesc-
id.pt

ABSTRACT

This paper describes an object-oriented abstraction for the problem of awareness management in Collaborative Virtual Environments (CVEs). The described abstraction allows for different types of awareness information and awareness management policies to be used. It is also described how the defined abstraction was used to support the awareness management policies of two demo CVEs application.

1. INTRODUCTION

Awareness is a very important concept in CSCW systems. As stated in [5] "awareness is an understanding of the activities of others, which provides a context for your own activity". Collaborative Virtual Environments (CVEs) are networked virtual environments used to support collaborative work. Users are represented graphically within the environment and can perceive other users actions through their graphical representation. In these systems, awareness information is all the information about existing objects and users within the system. This information includes users and objects graphical representation, the sounds produced by users, and users actions. These systems also have the characteristic of aiming to support a large number of simultaneous users in the order of tens of thousands. In the presence of a large number of users the amount of information that must be processed by each user can be overwhelming. As such, it is usually necessary to manage the amount of information that must be processed by each user. This is called awareness management. The goal of awareness management is to allow each user to only process the information that is relevant for him. Some of the existing systems use awareness management as a mechanism for reducing network bandwidth and increasing the system's scalability, while others use awareness management to promote user collaboration by using it to scope user interaction. The work described in this paper was done in the context of the MOOSCo (Multi-user Object-Oriented Virtual Environments with Separation of Concerns) project. MOOSCo proposes a

software engineering separation of concerns approach for the development of Multi-user Virtual Environments. For each of the different aspects of these systems, different concerns are identified. The system functionality is obtained by composition of the solutions defined for each of the concerns. In this paper is described an object-oriented abstraction for the awareness management concern. Further details about the MOOSCo approach can be found in [1, 2]. The rest of this paper is structured as follows. Next section describes the related work. Afterwards, we present the awareness management abstraction. The Implementation section describes two demo applications and their use of the abstraction to support different awareness management policies. The paper finishes with conclusions and future work.

2. RELATED WORK

Awareness management is a very important issue in CVEs. It is used as a mechanism of regulating the amount of information each user must process. Awareness management helps collaboration between users, by suppressing all the awareness information about users and objects that are not relevant for the current users' collaborative task. Awareness management also has an important role in the scalability of this kind of systems. By limiting the amount of information that must be processed by each user, awareness management can be a very effective mechanism for reducing resource usage, like network bandwidth and computer processing power. Given the importance of awareness management, different policies have been used depending of the systems requirements and goals. In RING [6] users are only aware of the objects they can see. This policy is adequate for environments that contain several visual barriers such as walls and doors, but it performs badly in densely populated open space environments. SPLINE [3] partitions the environment in spatial regions called locales. Each user is aware of all the objects in the current locale and in the immediate neighbours. The partitioning of the environment is a very flexible mechanism for structuring the virtual environment. In SPLINE each locale defines its own independent coordinate system. The virtual environment results from the connection of several locales. Each connection between two locales defines a 3D transformation that describes the relations between the locale's coordinate systems, which allows the creation of non-Euclidean environments. This approach also eases the extension of the environments, since it is only a matter of defining new locales and connecting them to the existing ones. Finally, the locales approach also provides an effective mechanism for controlling awareness. Allowing the awareness of the adjacent locales gives users the notion of spatial continuity, in-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VRST'01, November 15-17, 2001, Banff, Alberta, Canada.
Copyright 2001 ACM 1-58113-427-4/01/0011 ...\$5.00.

creasing at the same time the system scalability. Unfortunately, SPLINE only provides this built-in policy. There are some situations where different policies could be more useful. For instance, in some situations one could be interested only in the current locale, and in other situations, in the current locale and perhaps a small subset of the adjacent locales. NPSNET [9] divides the environment in fixed size regions called cells and each user defines an area of interest through an aura. Users are only aware of those cells, which their aura intersects with. The size and shape of the cells were chosen taking into account the target application domain, military simulation. The goal of awareness management in this system is to improve system scalability and to support a large number of simultaneous users. MASSIVE-1 supports the spatial model of interaction [4]. In this model users defined auras and interaction between two users can only happen when the user's auras intersect. The model also uses the concepts of focus and nimbus to compute the awareness level that a user can have of another user or object. The focus represents an observing object's interest in a particular medium. The nimbus represents an observed object's projection in a particular medium. The awareness level that an object A has of an object B in a particular medium M is a function of A's focus and B's nimbus in M. In MASSIVE-2 [8] the spatial model of interaction was extended with the third-party objects concept that represents objects that can affect other objects and users awareness levels by changing their values of aura, focus and nimbus. The spatial model of interaction is perhaps one of the most complex and flexible awareness management model. It is suitable for controlling user interaction in large-scale virtual environments. However, the model is too much focused on the spatial aspect of this systems, making it difficult to manage awareness using semantic or organisation considerations. In MASSIVE-3 [7] an extension to the SPLINE model was adopted. In this extension the environment is also divided into locales and the locales can be connected through boundaries. Each locale can have several aspects each representing a certain type of awareness information. The awareness management is performed, by selecting the locales and correspondent aspects that are relevant to a particular user. The system allows the programmer to select, or even adapt, the policy for selecting the relevant locales and aspects. Since locale's aspects can be arbitrarily defined it is possible to support awareness management taking into account organisational associations between the objects of an environment. Due to the existence of different application requirements it is necessary to support different awareness management policies.

All the mentioned systems only offer support for a particular policy or family of policies. Only MASSIVE-3 allows some degree of adaptation, by letting the programmers to choose the policy for selecting the relevant locales for a particular user. However, the adaptation is confined to a particular kind of awareness policy, based on locales and aspects. It is not possible to use different policies. This problem is normally due to the lack of proper design abstractions that are, not only, able to solve the problem at hand, in this case awareness management, but also able to support the several variations that exist for the problem solution. To cope with this problem, this paper presents an object-oriented awareness management abstraction that is flexible enough to support different awareness management policies. Instead of trying to provide a one-size-fits-all solution for awareness management, the defined abstraction allows for different policies to be defined, and for programmers to choose which is the most appropriate policy for the application being developed.

3. AWARENESS MANAGEMENT ABSTRACTION

This section describes an object-oriented abstraction for awareness management in CVEs. The goal of the proposed abstraction is not to define a generic model for awareness management that can be used for all the kinds of CVEs systems. Instead the abstraction aims to provide a common framework upon which different solutions for awareness management can be built. To be able to achieve this goal it is necessary that the abstraction is flexible enough to capture the several variations that exist in solutions for the problem of awareness management. The description of the Awareness Management abstraction is divided in four sections. Firstly, there is a general description of the problem that must be solved. Secondly, different variations that any flexible solution for awareness management should support are described. Afterwards, it is described the solution, i.e. the proposed object-oriented abstraction for awareness management. The solution presentation consists of: a description of the structure and the elements that are part of the solution; a description of how those elements collaborate to solve the problem at hand; and a description of how the abstraction supports the variations that were identified.

3.1 Problem

One of the issues to consider for awareness management is: what type of information must be considered for awareness management purposes? For instance, the sound produced by a certain user A can be used as awareness information. In this case a user B will be aware of A when he hears him. The geometric appearance of objects and users can also be considered as awareness information. A user becomes aware of objects and users from the moment he/she sees them. Even the user actions can be used as awareness information. For example, an user A may be able to see an user B but be unable to understand the actions he/she is executing due to the distance between them. After approaching user B, user A may then be aware of the actions user B is executing. Once the information used for awareness management is defined, it is necessary to define how the users declare their interest in certain types of information. Although there might exist different types of awareness information, a certain user may, at a certain moment, only be interested in a particular type. Once the awareness information and mechanisms by which the users express their interest are defined, it is necessary to guarantee that each user only receives the information that is relevant for him. There are several policies to manage who should receive a certain type of information. For instance, the proximity to the information source may be a way to define who may receive it. Moreover, the existence of visual barriers may be used to determine who shouldn't receive certain visual information. Whatever the management policy chosen, it is important that users understand the management policy being used so they can behave in order to control the awareness they have from the remaining users and objects.

3.2 Variations

A solution for Awareness Management must support the following variations:

Awareness Information Definition. It is necessary to allow the definition of different types of awareness information in accordance with the application's requirements. The choice of awareness information can be determined by the applications functional requirements, e.g., collaboration, or by non-functional requirements, e.g., the system ability to support a large number of simultaneous users.

Different Awareness Management Policies. Different awareness management policies represent different ways of managing the

awareness information that should be received by each user. Each application should be able to choose which awareness management policies are the most appropriate, and should also be able to define their own awareness management policies, that take into account the applications specific requirements.

3.3 Solution

The main characteristic of the proposed solution for the awareness management concern, is the separation between the elements that are subject of awareness management, awareness information sources and entities interested in consuming that information, and the policy responsible for managing the information that is consumed by each of the interested parties. This separation allows changing the management policy used, independently from the awareness information and the behaviour associated with handling that information. This separation also allows the use of different types of awareness information, independently of the awareness management policy used.

3.3.1 Structure and Participants

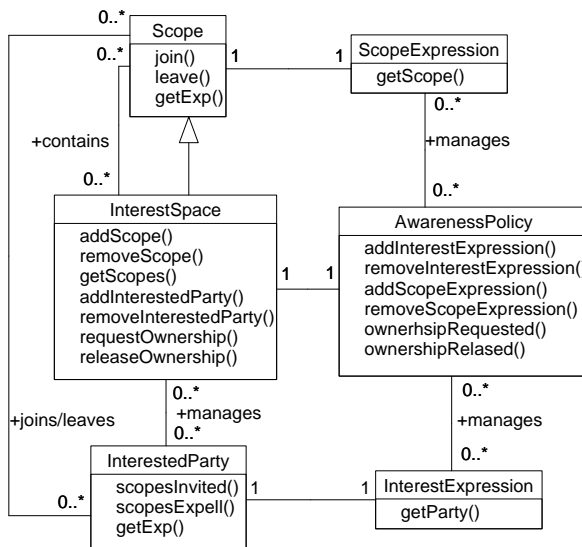


Figure 1: Class diagram of the Awareness Management abstraction structure.

The abstraction Awareness Management has the following participants:

- *Scope*. Represents a source of awareness information, more precisely the scope of a certain type of information, for instance the audio scope of an user.
- *ScopeExpression*. Represents expressions that describe a certain information scope. Each Scope has a scope expression associated that describes the type and the scope of the information it represents.
- *InterestedParty*. Represents an entity interested in a certain type of awareness information. It must be associated with a scope to have access to that information. The association and dissociation of interested parties to scopes is managed by the awareness management policy.
- *InterestExpression*. Represents the interest of an interested party in a certain type of awareness information.

- *InterestSpace*. Represents a space that contains several scopes for the same type of awareness information. Interested parties are registered in interested spaces indicating their interested expressions. An interested space can also play the role of a scope. In this way it is possible to define hierarchical interested spaces that use different awareness policies at each hierarchical level. The awareness management of the whole structure results from the awareness management of its constituent parts.

- *AwarenessPolicy*. Represents an awareness management policy. A policy is responsible for determining which interest expressions match scope expressions. Every time a match is detected, the policy informs the correspondent interested party that is should be associated with the scope. When the match ceases the policy informs the interested party to dissociate itself from the scope. Normally, the scope and interest expressions are dependent of the awareness management policy used. However, the interested parties and the information scopes are independent of the expressions being used. The association between interested parties and scopes, and interest expressions and scope expressions are created before registering the scopes and interested parties in a interest space. The choice of which interest expressions and scope expressions to use depends of the interest space' awareness policy.

3.3.2 Collaborations

The Awareness Management abstraction defines three types of collaborations: (1) register/unregister scopes in a interest space; (2) register/unregister interested parties in a interest spaces; and (3) association/dissociation of interested parties with scopes. A scope is registered in an interest space through its `addScope` operation. An `InterestSpace` obtains the scope expression from the scope, operation `getExp`, and registers it in the awareness management policy. To unregister a scope from an interest space it is necessary to invoke its `removeScope` operation. This operation will remove the correspondent scope expression from the space's awareness management policy. As a consequence, all the interested parties associated with this scope will be dissociated from it. For an interested party to be registered in a certain interest space it is necessary to invoke the operation `addInterestedParty` in the corresponding `InterestSpace` instance. Afterwards, the `InterestSpace` obtains the interest expression from the interested party, operation `getExp`, and registers it in its awareness management policy, `addInterestExpression` operation, so that it can manage the intersections with the existent scope expressions. An interested party unregisters from an interest space invoking the `removeInterestedParty` operation on it. Afterwards, the interest space invokes the `removeInterestExpression` operation in the awareness management policy to unregister the correspondent interest expression. As a consequence the interested party is informed to dissociate itself from all the scopes it was associated with.

Figure 2 shows the collaboration diagram for the association between an `InterestedParty` and one or more `Scopes`. The awareness management policy is responsible for detecting matches between interest expressions and scope expressions. The collaboration is initiated by the awareness management policy every time an interested party or scope is registered/unregistered from an interest space, or, as shown in Figure 2, when it detects a match between an interest expression and one or more scope expressions (1,2,3). The way matches are detected is policy's implementation specific. Every time one or more matches are detected, the pol-

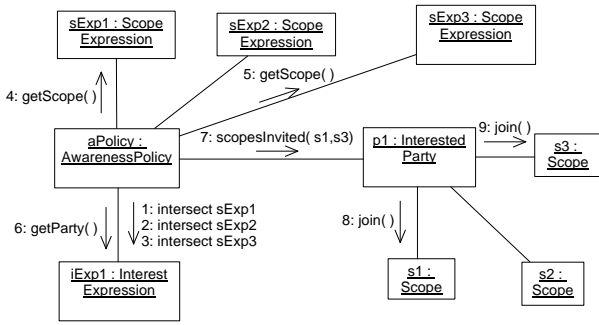


Figure 2: Collaboration diagram for the association between an InterestedParty and one or more Scopes.

icy informs the InterestedParty associated to the interest expression, to associate itself to the respective, operation `scopesInvited` (4,5,6,7). The InterestedParty associates itself with the Scopes invoking the operation `join` in each relevant Scope, (8,9). The behavior of the association between InterestedParty and a Scope depends of their implementation. The dissociation between an InterestedParty and a Scope is handled in a similar way to the association. Figure 3 shows the collaboration's diagram for the dissociation between an InterestedParty and one or more Scopes. In this case the policy identifies that one more Scopes are no longer relevant for the InterestedParty and informs it to dissociate itself from then.

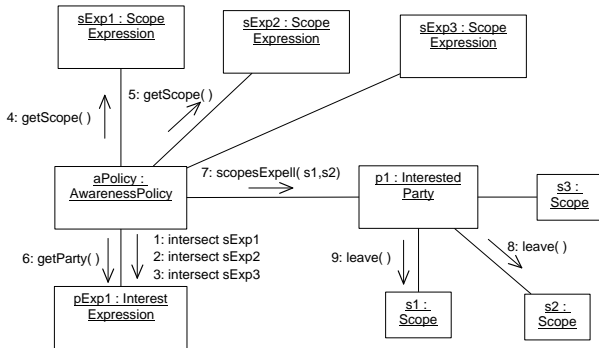


Figure 3: Collaboration diagram for the dissociation between an InterestedParty and one or more Scopes.

There is another type of collaboration that is a particular case of an association/dissociation of an interested party to a scope, and that occurs when this scope is also an interest space, i.e. a sub-interest space. Figure 4 shows the collaboration's diagram for the association of an interested party with a sub-interest space. The figure represents a interest space, `space2`, nested inside the interest space, `space1`, and an interested party `p1`, registered in the space `space1`. The awareness management policy of `space1` informs the interested party to associate itself to the subspace `space2`, operation `scopesInvited` (1).

The interested party associates itself with the space in the same way it associates itself with a Scope through the `join` operation, (2). The result of the association between an interested party and a sub-interest space is similar to the registration of an interested party in an interest space, i.e. the interest space will manage its associations with the existent scopes. The only difference is that

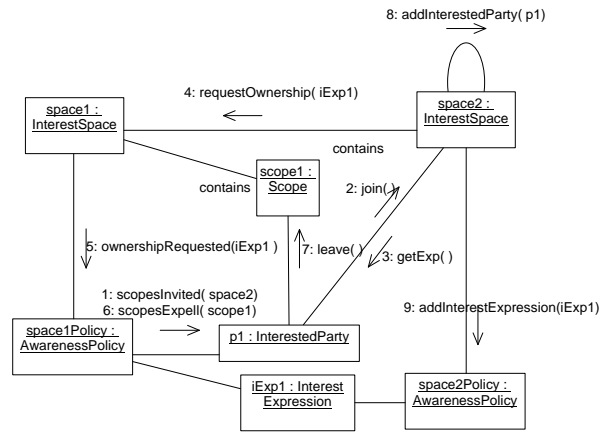


Figure 4: Collaboration diagram for the association between an InterestedParty and a sub-interest space.

a sub-space may request ownership of the interested party. In the case shown in Figure 4, the sub space requests the ownership by invoking the operation `requestOwnership` in the parent space, (4). As a consequence, the interested party is dissociated from every scopes managed by the parent space, except from the sub-space, (5,6,7). Finally, after obtaining the ownership, the interested party is registered in the nested space `space2`, so that the correspondent awareness management policy may manage what scopes from the nested scope should the interested party be associated with, (8,9).

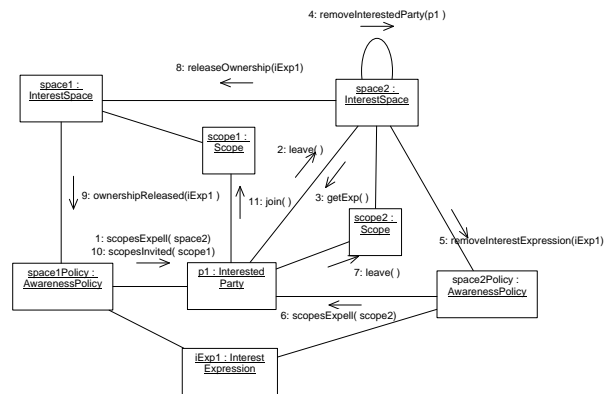


Figure 5: Collaboration diagram for the dissociation between an InterestedParty and a sub-interest space

The dissociation between an interested party and a sub-interest space is similar to the association. The only difference is that when the interested party is dissociated from the sub-interest space, the space releases the ownership of the interested party. This will allow it to associate itself to the scopes of the parent space. Moreover, the interested party is dissociated from all the scopes of the subspace. Figure 5 shows the collaboration diagram for the dissociation between an interested party and a sub-interest space.

3.3.3 Expressiveness

The abstraction Awareness Management supports different awareness management policies independently of the type of awareness information. The classes `InterestedParty` and `Scope` repre-

sents, respectively, the parties interested in consuming awareness information and the sources of awareness information. Different types of awareness information are represented by different specialisations of these classes. The class `AwarenessPolicy` represents an awareness management policy.

The separation between the awareness management policy and the types of awareness information is obtained through the classes `InterestExpression` and `ScopeExpression`. For each particular policy, these classes describe the interests of a particular `InterestedParty` and of a particular `Scope`. The solution also allows for hierarchical organization of interested spaces. Each interested space in the hierarchy has its own awareness management policy. The resulting awareness management policy results from the composition of all the policies of all the interest spaces in the hierarchy.

4. IMPLEMENTATION

The *Awareness Management* abstraction described in this paper was implemented in JavaTM as an object oriented micro-framework. This micro-framework is part of a larger framework called MOOSCo that supports the development of CVEs. The MOOSCo framework was developed using a separation of concerns approach. For each concern an abstraction was defined and implemented. The support for CVEs was obtained composing each of the concern's implementations. At the present moment, there are abstractions for replication, awareness management, distributed communication and object interaction. By supporting different sets of concern's compositions it is possible to support CVE's with different capabilities, enabling in this way incremental development. For instance scoped interaction is obtained by composing awareness management with object interaction. Adding replication and distributed communication a distributed and replicated virtual environment is obtained. Partial replication of the environment is obtained composing replication and awareness management. Depending how the MOOSCo framework is instantiated (which composition the programmer chooses) the functionality of the CVEs will be different. One of our goals is to promote incremental development by allowing programmers to change the instantiated composition, starting from the simpler ones and moving incrementally to the more complex ones.

In the remainder of this section we will present two applications developed using the MOOSCo framework that focus on awareness management.

4.1 Conference Table

This application consists of 3D chat distributed virtual environment. Users can move within a 3D environment and can interact with each other by text-based communication. This application was developed to demonstrate a particular usage of awareness management that uses nested interest spaces. In the environment there is a conference table, around which users can sit. The table functionality only deals with managing the number of available sits, allowing users (or software agents) to sit at the table only if there is a free sit. Awareness management is used to control the interaction between the users sat at the table and the remaining users of the environment. Users at the table cannot interact (chat with) users outside the table. However, depending of the awareness configuration, users outside the table may or may not "hear" what the users in table are saying.

To implement the awareness management for this application, specializations of `InterestedParty` and `Scope` were used: `InteractionInterestedParty` and `InteractionScope`. These specializations define awareness information as the interaction between users and objects (the act of sending messages be-

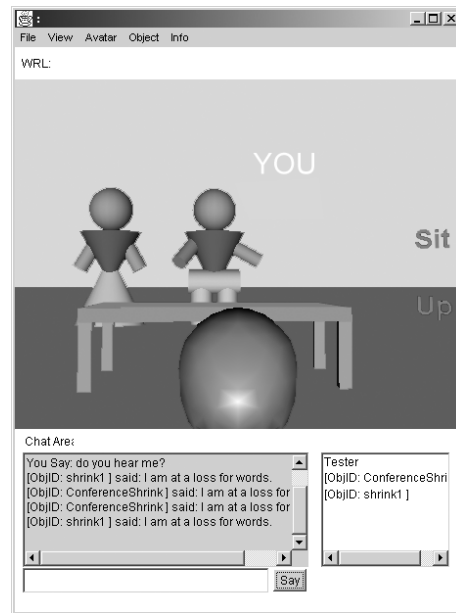


Figure 6: Two bots and a user interacting while sitting on a conference table.

tween users, or objects). A user can only receive messages from another user if he has joined the user's interaction scope. Each user defines an interaction interested party indicating that they want objects to interact with him; and an interaction scope indicating it is willing to interact with other users and objects. These specializations are in fact one of the concerns' compositions supported by the MOOSCo framework: the composition between the object interaction concern, that deals with interaction between objects, and the awareness management concern. This composition allows to support scoped interaction between objects controlled by a particular awareness management policy that determines when users should join others users interaction scopes.

To be able to scope the interaction between the users at the table, the ability of the *Awareness Management* abstraction to support nested interest spaces, two interest spaces were used. One space represents the environment itself, and the other represents the conference table and is nested inside the first one. As described before, the nested interest space plays the role of scope in the parent interest space, in this case, the conference table's scope.

Each one of the interest spaces uses an awareness policy to manage the associations/dissociations of the registered interested parties and scopes. For the environment's interest space, an aura based awareness policy was used. This policy defines `InterestExpressions` and `ScopeExpressions` as auras centred at each user's position. When an `InterestExpression` of a user A intersects a `ScopeExpression` of a user B, user A joins user B's scope. When users approach the table and enter the table's aura their interest expression joins the table's scope, and is inserted in the table's interest space. For the table's interest space a simple awareness policy was used that associates every registered interest party to every registered scope. This enables users to hear other users sat at the table. To enable the users to hear the new user, the user's scope is also inserted in the table's interest space. In this way all users sat at the table (whose aura is inside the table's aura) can hear each other. Enabling or disabling users outside the table to ear or talk with users at the table, is accomplished using the

“take ownership” functionality described in 3.3.2. If the table’s interest space is configured to take the ownership of each registered interested party, then when a user joins the table’s scope leaves the scopes of all users outside the table, and hence ceases to hear them. If the table’s interest space is also configured to take the ownership of each registered scope, then the interested parties of the users outside the table will be forced to leave the scopes of the users at the table, and hence will not be able to hear them.

4.2 Distributed Art/Life

Distributed Art/Life is MOOSCo’s adaptation of a previous standalone application. The original application was developed to demonstrate emergent behaviour theories on aggregate moving. The application’s main goal is to simulate a world inhabited by living creatures. Each creature reacts to its environments according to a set of predefined behaviours. These creatures may have one of two roles: predators or preys. The latter tend to aggregate in flocks, whereas predators will predominantly act alone. The *boids* model, used in Art/Life, was first defined by [10, 11]. This model tries to describe a population’s behaviour through the individual behaviours of its elements. It was developed to describe the coordinated group movement of flocking birds, schools of fish or herds. To achieve the desired emergent behaviour it relies on each individual’s perception of its surroundings.

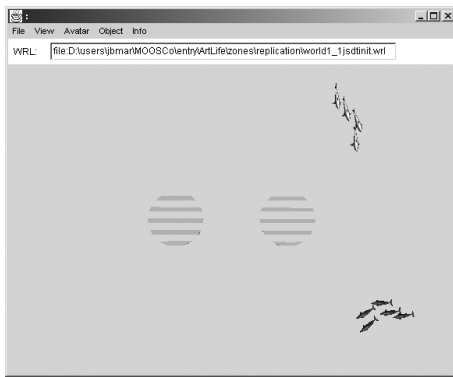


Figure 7: Two species, simulated by different users and unaware of each other

The Distributed Art/Life application supports a distributed simulation of the *boids* model in a 3D virtual environment. In this application each participant can simulate one or more *boids*. Each participant also replicates the other participant’s *boids*. As a result its *boids* perceive the other participant’s *boids* allowing the emergent behaviour to develop.

This is a very interesting subject for awareness management as one can affect the population’s emergent behaviour by using awareness management to control the perception that individuals have of each other. For this application two different specializations of Interested Party and Scope were used:

- *InteractionInterestedParty and InteractionScope*. Are used to scope interaction between *boids* in a similar manner as described before, a *boid* can only interact with another *boid* if its *InteractionInterestedParty* has joined the other *boid*’s *InteractionScope*.
- *ReplicationInterestedParty and ReplicationScope*. Are used in conjunction with replication to support partial replication. Each participant of the distributed

simulation has a *ReplicationInterestedParty* and *boids* are associated to *ReplicationScopes*. When a participant’s *ReplicationInterestedParty* joins a *ReplicationScope* all the *boids* associated to that scope are replicated in the participant’s machine.

In addition to the above specializations two awareness management policies were used/defined:

- *aura based policy*. This policy is the same awareness policy used in the *Conference Table* application.
- *flock based policy*. This policy defines *ScopeExpressions* as a string that identifies a flock that a scope belongs to and defines *InterestedExpression* as a set of flocks an interested party is interested in. An interested party joins a scope when the scope’s flock is one of the flocks described in its interested expression.

By using different combinations of the specializations described above the resulting awareness management has different effects in the *boids* population behaviour:

- *InteractionInterestedParty and InteractionScope with aura based policy*. In this case the interaction between *boids* is scoped. Each *boid* can only interact with the *boids* whose scope’s aura intersect its interest aura. In this case the size of the auras will influence the level of flocking. If auras are too small only *boids* that are sufficiently near each other will flock. If auras are big enough then the effect of awareness management may not be felt and all the *boids* will flock.
- *InteractionInterestedParty and InteractionScope with flock based policy*. In this case each *boid* will only flock with *boids* whose scope have the same flock identification. Each *boid*’s interest expression contains only a flock identification, the *boid*’s flock. With the exception of predators that are interested in all *boids* of all flocks which means they can pursue any *boid* since they are aware of all of them. In this way there will be as many different flocks as the flocks IDs defined.
- *ReplicationInterestedParty and ReplicationScope with flock based policy*. In this case the awareness management is being used to support partial replication. *Boids* are organized in flocks that represent different *ReplicationScope*. Each participant defines a *ReplicationInterestedParty* with an interest expression containing the identification of the flock it is interested in (typically, the flocks for which he is simulating some *boids*). In this situation each participant only replicates the *boids* belonging to the flocks he is interested in. Here the flock organization could be used to separate *boids* into species (a flock would represent a species). The problem with this combination is that since there is no scoped interaction, if a participant simulates *boids* from different flocks (species) then all *boids* will eventually be aware of each other which will cause them to flock together.
- *Both InterestedParty and Scope specializations with flock based policy*. This case uses both scoped interaction and partial replication, allowing distributed simulation of different flocks and at the same time allowing each participant to only replicate the *boids* belonging to the flocks he is interested in.

5. CONCLUSIONS

In this paper an object-oriented abstraction for awareness management was proposed. This abstraction was developed under a separation of concerns approach for the developing of Collaborative Virtual Environments. In this approach, abstractions were defined for each of the different concerns of these systems. The abstraction described in this paper provides a generic solution for the problem of awareness management. It allows different types of awareness information and awareness management policies to be used. At the present, abstractions were also defined for object interaction, distributed communication and replication concerns [2, 1]. There is a JavaTM implementation of the defined abstractions and their compositions. In this system the awareness management abstraction was composed with the object interaction abstraction to support scoped interaction and with replication and distributed communication abstractions to support partial replication of the virtual environment.

This paper also described two demo applications that showed how the defined abstraction implementation can be used to define different awareness management policies and can be used to manage different types of awareness information. It was also shown how the same application can easily use different awareness policies.

Besides the awareness management policies described in this paper, other policies, like the ones found in systems like SPLINE [3] or MASSIVE-3 [7] can be define using the described abstraction. At the present moment only the SPLINE's locale-based policy and a grid based policy similar to the one in NPSNET [9] are implemented.

As future work other abstractions will be defined that support solutions for other concerns like persistence and authentication. There is much to be gained by identifying those abstractions. In the case of CVEs systems the definition of proper abstractions will allow to describe and extend the existent knowledge base, allowing at the same time to reuse and reason about the existent solutions

6. ACKNOWLEDGMENTS

This work was partially funded by Fundao para a Cincia e Tecnologia, Praxis/ C/ EEI/ 33127/ 1999 MOOSCo.

7. REFERENCES

- [1] M. Antunes, H. Miranda, A. R. Silva, L. Rodrigues, and J. Martins. Separating replication from distributed communication: Problems and solutions. In *International Workshop on Distributed Dynamic Multiservice Architectures (DDMA 2001)*, pages 103–108, Phoenix, Arizona, USA, April 2001. IEEE.
- [2] M. Antunes and A. R. Silva. Using separation and composition of concerns to build multiuser virtual environments. In *6th International Workshop on Groupware (Criwg'2000)*, pages 68–76, Madeira, Portugal, October 2000. IEEE.
- [3] J. Barrus, R. Waters, and D. Anderson. Locales: Supporting Large Multiuser Virtual Environments. In *IEEE Computer Graphics and Applications*, pages 16(6):50–100, November 1996.
- [4] S. Benford and L. Fahl. A Spatial Model of Interaction in Large Virtual Environments. In *Proceedings ECSCW'93*, pages 13–17, Milan, Italy, September 1993.
- [5] P. Dourish and V. Bellotti. Awareness and coordination in shared workspaces. In *In Proc. Computer Supported Cooperative Work (CSCW '92)*, 1992.
- [6] T. Funkhouser. Ring: A Client-Server System for Multi-User Virtual Environments. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 85–92. ACM SIGGRAPH, March 1995.
- [7] C. Greenhalg, J. Pubrick, and D. Snowdon. Inside MASSIVE-3: Flexible support for data consistency and world structuring. In *Proceedings of the 3rd International Conference on Collaborative Virtual Environments (CVE'2000)*, San Francisco, California, USA, September 2000. ACM Press.
- [8] C. Greenhalg and S. Benford. Boundaries, Awareness and Interaction in Collaborative Virtual Environments. In *Proceedings of the 6th Workshop on Enabling Technologies (WET-ICE '97) Infrastructure for Collaborative Enterprise*, Cambridge, Massachusetts, USA, June 1997. IEEE Computer Society Press.
- [9] M. Macedonia, M. Zyda, D. Pratt, D. Brutzman, and P. Barham. Exploiting Reality with Multicast Groups. In *IEEE Computer Graphics and Applications*, pages 15(5):38–45, September 1995.
- [10] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proceedings of the SIGGRAPH '87 Conference*, pages 21(4):25–34, Anaheim, California, July 1987. ACM SIGGRAPH.
- [11] C. W. Reynolds. Steering behaviours for autonomous characters. In *Game Developers Conference*. Miller Freeman Game Group, 1999.