Specifying Consistency Requirements for Massively Multi-User Virtual Environments

Laura Itzel, Richard Süselbeck, Gregor Schiele and Christian Becker
University of Mannheim
Mannheim, Germany
{ laura.itzel | richard.sueselbeck | gregor.schiele | christian.becker }@uni-mannheim.de

***Abstract*—In this paper we discuss that the specific consistency and latency requirements of world state updates in Massively Multi-User Virtual Environments (MMVEs) are determined by several characteristic factors of the virtual world, i.e. their *Interaction Context*. As this context is highly application-dependent, MMVE developers require a simple specification scheme which allows them to specify these requirements for their specific Interaction Contexts. In this paper we present such a scheme, which we believe represents an important step towards dynamic and flexible consistency management for MMVEs.**

I. Introduction

Consistency is one of the key challenges of Massively Multi-User Virtual Environments (MMVEs) where a massive number of users interact with each other in a large-scale simulated world. Ideally all users should always have the same (i.e. consistent) view of the world's state. However, any action performed by a user should be reflected in his view instantaneously. Clearly, both of these requirements cannot be fulfilled at the same time in a distributed system, due to network delay and bandwidth restrictions. Therefore, the right balance of consistency and responsiveness is important.

This balance is highly situation-dependent [1]. For example updates resulting from a virtual trade where two users exchange a virtual good for money, a low responsiveness is tolerated but consistency needs to be high. For low-value items, lower consistency may be accepted, while higher responsiveness is required in a battle, when users exchange items such as health potions. Similarly, movement updates often require low consistency, but in the context of a virtual race, the consistency requirements for movement updates increase.

Thus, the consistency requirements for each update are dependent on the characteristics of the corresponding action and the virtual world. We call this the *Interaction Context* of the update. All updates classified by a specific context have the same consistency and latency requirements. A flexible consistency management must therefore provide a scheme to the application developer to i) determine the set of Interaction Contexts relevant to his application and to ii) specify the consistency and responsiveness requirements based on this set of Interaction Contexts.

In this paper, we provide such a scheme. To do so, we first define the concept of an Interaction Context by providing its characterizing parameters. Furthermore, we propose a set of metrics which allow the developer to specify the requirements for each Interaction Context. The resulting specification then serves as the input for our flexible, yet generic consistency management system for MMVEs which we are developing in the Peers@Play project [2]. Based on this specification, this system can then optimize the propagation of updates [3][4].

II. Interaction Context

We define the *Interaction Context* of a given update in an MMVE as the set of characteristics of the corresponding interaction and the virtual world, which influence the consistency and responsiveness requirements of the update. We have identified the following characteristics:

Update Type - The update type refers to the specific action the update represents. For example this could be *Position Update* for avatar movement or *Trade* for handing over a virtual good to another avatar.

Affected Entities - This refers to the entities affected by the update, e.g. avatars or objects involved in the interaction. This includes both the entity that initiates the interaction as well as those that are affected by its effects.

Dependent Interactions - This denotes the update's dependence of and influences on other interactions than its own. We must both consider existing interactions, e.g. an ongoing fight or trade, as well as potential future interaction, e.g. whether the outcome of the update can influence any interactions which have not yet taken place.

III. Consistency Metrics

Based on the Interaction Context of an update, the application developer can specify the consistency and latency requirements in his application. Therefore, we have identified three *Consistency Metrics* for specifying those requirements of an update in a given Interaction Context:

Inconsistency Tolerance - Due to network latency it is not feasible for each user to perceive the exact same world state at the same time. Therefore, this metric enables the application developer to specify the minimal requirements for each state update in terms of the maximum state divergence tolerated. The state divergence can for example imply the distance between an entities' actual and perceived position.

Interactivity - This metric indicates an update's requirements regarding responsiveness, i.e. how much latency the action associated with the update can tolerate. The longer it takes for an action to be processed by the consistency

management, the less responsive the application appears to the user. Using this metric, the application developer can specify the maximum acceptable amount of time until the state change resulting from the action appears to the user.

Note that the interactivity of an update depends on its tolerance for inconsistencies, but requiring high interactivity does not indicate if and how the state is allowed to diverge. However, due to this dependency certain requirements cannot be fulfilled in specific situations. The consistency management must then notify the application.

Priority - Priority enables the developer to specify a possible relaxation of the consistency requirement, in case the system cannot sustain the desired level of consistency for all updates. The consistency management can use the priority to determine for which updates it can reduce the provided consistency first. For example, the consistency for certain movement updates would be relaxed first, while updates concerning virtual trades would be treated with the highest possible level of consistency as long as possible.

Using these three metrics we can dynamically adapt the synchronization of an MMVE in such a way that the user experience is least affected, while maintaining minimum consistency requirements.

IV. Specification Examples

To use our specification scheme, an application developer has to i) identify the set of Interaction Contexts relevant for his application and ii) specify the consistency and responsiveness requirements using the Consistency Metrics. At runtime the consistency management then detects the Interaction Context for each update and provides the required level of consistency by fulfilling the associated metrics.

In the following we present an example to illustrate the usage of our concepts in an MMVE gaming application. We specify the Interaction Contexts of three different movement updates and classify them according to our metrics.

Context 1 represents a position change of a single avatar. The application developer specifies the *Update Type* as *Position Update*, the moving avatar as the only *Affected Entity* and identifies no *Dependencies* in this context. An example update which could be classified to this context is the movement of an avatar A with no other entity nearby to interact with. Thus a good classification of this context is high *Inconsistency Tolerance*, low *Interactivity* and low *Priority* values.

Context 2 also is of type *Position Update*, but with two avatars as *Affected Entities*. In contrast to the first context, the updates in this context may influence or may be influenced by other interactions. For example, if avatars A and B are standing close enough to be able to interact, the exact position of A would be of interest for B. This context can be classified with a medium *Inconsistency Tolerance*, medium *Interactivity* and medium *Priority* as no ongoing interaction is actually influenced by the movement.

Context 3 again is of type *Position Update* with two affected avatars, but the position change will likely have an impact on other interactions. For example, if avatars A and B are fighting with each other, B inflicting location-specific damage on A, it would be critical for B to instantly see the exact position of A. A good classification for this context is low *Inconsistency Tolerance*, high *Interactivity* and medium *Priority* values.

Note that this represents an example subset of Interaction Contexts and their associated metrics for a specific application. Due to the application-dependent nature of consistency requirements, both contexts and metrics can be very different in other applications.

V. Related Work

Most other proposals for consistency management in MMVEs do not consider this dependencies on the specific context of an update. Existing approaches mostly specify the consistency requirements based on the specific object (e.g. [5], the current location of the object (e.g. [6]) or the type of interaction (e.g. [7]). In [8] the requirements are defined based on scenarios, which combine several interactions.

In contrast, we develop a flexible yet generic consistency management system for MMVEs which supports varying consistency and responsiveness requirements based on the context of the interaction [3][4].

VI. Conclusion and Future Work

In this extended abstract we have introduced the concepts of Interaction Context and Consistency Metrics. In combination they provide a specification scheme for the consistency and responsiveness requirements of updates in MMVEs. This scheme can be used by developers to provide their application-specific requirements for a consistency management, which uses them to dynamically adapt the MMVEs update propagation. Our future work includes detection of the interaction context at runtime, integration into our existing consistency management followed by an evaluation of our concepts.

References

[1] W. Palant, C. Griwodz, and P. Halvorsen, "Consistency requirements in multiplayer online games," in *ACM SIGCOMM workshop on Network and system support for games (NetGames 06)*. New York, NY, USA: ACM, 2006.

[2] Peers@Play Project, "http://www.peers-at-play.org."

[3] L. Itzel, V. Tuttlies, G. Schiele, and C. Becker, "Consistency management for interactive peer-to-peer-based systems," in *Workshop on Distributed Simulation and Online Gaming (DISIO 10)*. ICST, Brussels, Belgium, Belgium: ICST, 2010, pp. 1–8.

[4] G. Schiele, R. Süselbeck, A. Wacker, T. Triebel, and C. Becker, "Consistency management for peer-to-peer-based massively multiuser virtual envirnments," in *1st International Workshop on Massively Multiuser Virtual Environments (MMVE 08)*, 2008.

[5] T.-C. LU, M.-T. LIN, and C. LEE, "Control mechanism for large-scale virtual environments," *Journal of Visual Languages & Computing*, vol. 10, no. 1, pp. 69 – 85, 1999.

[6] H. Schloss, J. Botev, M. Esch, A. Hohfeld, I. Scholtes, and P. Sturm, "Elastic consistency in decentralized distributed virtual environments," in *International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution (AXMEDIS 08)*, 17-19 2008.

[7] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "Von: a scalable peer-to-peer network for virtual environments," *Network, IEEE*, vol. 20, no. 4, pp. 22 –31, july-aug. 2006.

[8] C. Savery, T. C. N. Graham, and C. Gutwin, "The human factors of consistency maintenance in multiplayer computer games," in *16th ACM international conference on Supporting group work (GROUP 10*. New York, NY, USA: ACM, 2010, pp. 187–196.