

Requirements for Equator Record and Reuse Infrastructure

Steve Benford, Chris Greenhalgh, Adam Drozd,
Duncan Rowland, Martin Flintham

The Mixed Reality Laboratory
The University of Nottingham
Nottingham, NG8 1BB, UK
{sdb, cmg, asd, dar, mdf} @cs.nott.ac.uk

Introduction

Developing a flexible and generic record and reuse mechanism for mixed reality experiences is a complex and challenging task. In this position paper we break the problem down into a number of sub-problems, expressed as requirements and associated technical approaches and challenges. In particular, we:

- Identify requirements for a generic record and replay facility within the Equator infrastructure. These are ordered in terms of increasing sophistication both with regard to how flexibly recorded material can subsequently be reused as well as increasing difficulty of realisation.
- Identify the kinds of tools that we might wish to provide to users so that they can make use of this facility
- Identify some of the underlying technical challenges to be addressed in developing the facility and associated tools.

Background literature

The issues raised in this paper stem from our previous experience of developing record and replay mechanisms for collaborative virtual environments (most notably for the MASSIVE-3 system [14]) in combination with our more recent experience of developing Equator's Equip infrastructure.

In this section, we briefly consider the existing literature on record and replay mechanisms for collaborative virtual environments. Several virtual environments support some form of record and replay facility. In the CAVERNsoft system, recording of an avatar's movements and audio is possible as part of its general support for persistent virtual environments [12]. This facility has been used to create the Vmail system [10], a form of VR email, and to create guided tours within tele-immersion applications. As part of the COVEN project, the DIVE system was extended with event logging facilities that could completely record an entire virtual environment and the activity within it [4]. Although initially implemented to support the statistical analysis of patterns of user activity in relation to network traffic, this recording facility was subsequently extended with a replay facility to allow a previous

session to be recreated (although not within another live virtual environment). The MASSIVE-2 system also used system logging to support analysis of network performance in relation to user activity, for example analysing correlation of speech in relation to volumes of network traffic [9].

Multi-player 3D games employ record and replay techniques to show highlights of previous game-play, examples of which include FIFA soccer from Electronic Arts and the automobile game Driver from GT Interactive Software. The latter also allows players to edit together their own movies from recordings of their own actions. In a related area, recent work on 'virtualized reality' has developed techniques for capturing live action within a physical environment by analyzing recordings from multiple video cameras to produce a 3D graphical simulation [11]. Furthermore, a script language VRML History has been proposed to facilitate time navigation in WWW browser-based 3D-worlds [13].

Examples of related work which refers to temporal properties in multimedia applications, but not specifically to VR, include the Media Editor Toolkit (MET++) application framework which has methods for manipulation of temporal relations in multimedia presentations [1], the Command Stream Model (and associated TclStream implementation) which introduces a media stream of code fragments that is synchronised to conventional media streams or device controllers, allowing presentation with VCR-type playback [5], and the spatio-temporal model of Vazirgiannis et al. which provides a framework for composition and indexing of multimedia objects [16]. The Where We Were (W3) system is being developed to allow recently recorded video, and 2D sketches, to be incorporated into real-time activities such as brainstorming meetings [15], and the Virtual Meeting-room Service (VMS) presents visual representations of recorded histories from distributed collaborations [5]. The CHIMP framework considers collaborative authoring and editing of multimedia documents, which includes temporal specification of objects [3]. The World Wide Web Consortium has also adopted the Synchronized Multimedia Integration Language (SMIL) as a recommendation that deals with the temporal aspects of authoring multimedia productions [19].

Finally, in our own previous work on the MASSIVE-3 system [14], we introduced a flexible mechanism for recording activity in virtual environments, manipulating such recordings, and then accessing them as new content within other live virtual environments [7]. This included the ability to access recordings within recordings. At the heart of our mechanism was a technique for capturing all activities within a CVE, including the environment itself, multiple users' movements, speech (over real-time networked audio), and interactions with virtual objects, in such a way that this action could then be faithfully recreated at a later time. The resulting 3D recordings could be embedded within a live virtual environment and application developers and end-users were able to program and also directly manipulate the temporal, spatial and presentational relationships between a live environment and the recordings. Action within the live environment could take place around and within the display of the recorded activity, and the composition of the two could itself generate further recordings. We demonstrated a variety of applications of this mechanism including: quickly developing content for virtual worlds by having live actors improvise scenes; accessing virtual mail messages and flashbacks within live virtual worlds; and supporting both HCI and system-level analysis of CVE experiences [8].

High-level requirements of record and replay

We begin with general requirements for a generic Equator record and reuse facility.

Logging for analysis

Perhaps our most basic requirement is to be able to log system data for subsequent offline analysis, for example in order to be able to understand participants' patterns of activity and how they relate to system performance. This doesn't even require that an experience can be replayed. It does require however, that sufficiently rich information is captured. This will include a history of 'system state'. In a stateful system such as a tuple-space, this means capturing a complete history of operations that updated the state; in an event based system such as Elvin it means capturing a history of events; in a hybrid system such as Equip it means capturing both. Entries in the log have to be appropriately time-stamped.

Analysis will involve tools that are able to inspect the log files, generating different visualisations of the activity that occurred or loading them into other data analysis tools (e.g., SPSS and Excel), which may require appropriate data format converters.

Conducting a thorough analysis of system performance may require further information to be captured such as raw logs from sensors (e.g., GPS logs) and network traffic logs (e.g., produced by tools such as tcpdump). It may also be necessary to capture information describing system configuration (i.e., which processes were running where and were connected to which other processes).

Finally, in distributed systems there is the option to capture these various logs at different software components, especially where they are distributed across a network and so separated by time delays. This allows an analysis to take account of the fact that different components may have experienced events at different times or even in different orders. In systems that are designed to work in the face of regular disconnection (e.g., where clients continue to provide some level of experience even when not connected to a server) or that are designed around non-centralised architectures (e.g., that operate in a peer-to-peer mode), it is even more important to capture logs at different clients as there is no single component that can provide a canonical view of the experience.

Direct replay of an experience from start to finish

Another core requirement is to be able to replay an experience and review it as if live, but from the point of view of an external observer (i.e., where the observer is not part of the experience). In the simplest case, one component of the system – typically a central server – is considered to offer the canonical view of the experience and generates the log files – henceforth called a recordings – that are replayed (i.e., a simple replay doesn't account for the fact that different participants may have had different perspectives). Recording might be played back through a special viewing client, with the whole replay system potentially being a standalone application that can run independently of the original system (and probably in a much scaled down configuration, for example on a single computer).

Explicit queries over recording information

Rather than replaying recordings, some applications may wish to make explicit queries over the recorded material, e.g. of the form "what happened?", or "what were the values of these items during the specified period?". This is typical both of the

“semantic” perspective to events and recording in projects such as CoAKTing, and also to various Equator e-Science scenarios (e.g. the time-series of readings returned by the Antarctic device). Note that – unlike the typical analysis scenario – these queries may be asked within the ongoing and uninterrupted use of the system.

This may be regarded as falling outside the scope of this paper (whose primary scope is on replaying recordings), but is a concurrent and related requirement. It is not yet clear whether the same mechanism and facilities will be able to support ‘query’ as well as ‘replay’ or whether these introduce fundamentally different requirements.

Replay with a changing viewpoint

The observer will usually want to be able to take different perspectives on the replay, for example moving a virtual camera to specific locations or choosing to follow particular characters. In particular, a viewer after the event may wish to take quite different perspectives to the original participants. This is achieved by extending the replay client.

Replay with multiple simultaneous viewpoints

It may be useful to be able to view a single replay using multiple viewers, either to support collaborative review sessions or in situations where different clients are specialised to provide different views (e.g., one provides an overview of the environment and another a close up view of particular participants). This requires a framework for coordination and datasharing between the clients. One possibility is to use the distributed system that originally generated the recordings (e.g., Equip). This leads to the approach of re-injecting the recordings back into the original system as if they were live data (rather than having specialised standalone viewing tools), enabling developers to create and link clients in the usual way. However, it needs to be noted that interacting with the recordings (i.e., manipulating and changing the data that is injected back into the system) may not generally be possible without further extensions as this may break the causality between recorded actions. Although the recorded data may appear to be live, it is not – specifically, it is read-only.

Replay only selected parts of a recording

A further requirement may be to replay only selected parts of a recording. This may involve selecting only those objects that match specified patterns (e.g., replaying the actions of certain participants, the events that happened around a particular location, or omitting certain kinds of objects). This requires a way of specifying the material to be selected. It also raises further problems of dependencies between objects and causality (for example, what to do if the state of a selected object depends upon or affects an object that is not selected).

Another example of selection is temporal selection in which only certain moments from the recording are replayed (e.g., between specified begin and end times). This introduces a further technical issue – how to best initialise the starting state of the replay. One option is to replay from the beginning of the entire recording, but to inject all recorded events up to the desired start time as quickly as possible (i.e., not in real time). Another is to add checkpoints to the recording file – i.e., state snapshots (rather like keyframes in animations) that allow the system to directly begin replaying from a specific moment in the middle of the recording.

Control the timing of a replay

It may be desirable to be able to control the timing of a replay. One aspect of this is being able to jump – forwards and backwards – to specified moments in time. This raises the same issues of managing state as we saw with selecting moments in recordings, although with this requirement the mechanism may need to work with an immediate response where the ability to jump is under direct real-time user control.

A second aspect is being able to replay a recording both forwards and backwards. This introduces a further technical challenge because playing backwards is not simply a case of performing the recorded operations in the reverse order. For example, undoing the effects of an ‘additem’ operation involves deleting an item. In order to be able to rewind time it is therefore necessary to generate anti-operations (either on the fly or inserted into the recording itself).

A final aspect of controlling timing is being able to speed up and slow down the replay (either forwards or backwards in time). This requires an underlying mechanism to map between the timings of events in a recording and the time of the replay.

Playing and synchronising multiple recordings

A yet more sophisticated record and replay facility might enable multiple recordings to be played back at the same time in a mixed and synchronised way (analogous to the way in which multi-track audio recording systems enable separately recorded audio tracks to be mixed together). This involves being able to inject multiple recording files (including those from which material has been selected) into the replay system and synchronising their replay to a common clock. It may also involve relating the recordings to one another in other ways, for example in spatial ways where different recording files might be associated with different locations.

Replaying recordings as part of live experiences

A further potential requirement is to be able to replay recordings within an ongoing live experience. This means that the recorded data appears to be mixed in with the live data and that participants can continue to act around it. This would allow replaying scenes from the past as ‘flashbacks’ within a live experience or would enable participants to leave recorded messages for others to find and view later on.

A key constraint is that the recorded data cannot interact with the live data (otherwise there are potential problems with causality) and so in reality, the live and recorded events are quite separate, but only appear to be overlaid. Further questions concern whether and how components of the system (e.g., viewing clients) and end users are aware of what is live and what is recorded.

A further possibility here is to enable a live environment to contain a replay of itself as it was sometime ago in the past (in effect, creating a link between the live and recorded state of the environment).

Making nested recordings

A further requirement might be able to make new recordings from combinations of existing recordings and live action. For example, we might replay a recording, enact some new experience around this and then save the combination as a new nested recording. Similarly, it may be desirable to save selected and then mixed recordings as a new recording.

A key issue here concerns the structure of nested recordings: do they use an inline structure where the old and new events both appear as logged data within the recording or is there – possibly more flexibly – a structure of separate but interlinked recording files?

Replaying the complete system perspective

In cases where there is no single process that can capture the canonical state of the experience or where we may want to explore the relationship between divergent views of state due to delays, errors and disconnections, it may be useful to be able to replay the complete state of a system, i.e., recreate an experience as it would have been seen on all of the participating components. This may involve playing back multiple recording files from different devices in a synchronised way, combined with appropriate simulation of the original system configuration and network performance.

Synchronising with external media

Capturing a mixed reality experience, such as a game that takes place simultaneously on the city streets and online, may involve recording several different media types, including system state and more conventional media such as video. There is already a wide variety of tools for capturing and manipulating digital video, based on common standards, and our system state recording mechanism will need to interface to these. For example, we will need to cross index between times and events in video and system state recordings and to synchronise their capture and replay.

Interacting with recordings

We may wish that users can interact with recorded data when it is replayed. However, this raises challenging problems of causality and agency. For example, if a live participant collides with a recorded object then the subsequent recorded history for that object may no longer be valid, which in turn, will affect any other recorded objects that were – directly or possibly indirectly – affected by it. If the object in question was under human agency (e.g., was another participant), then reinstating the live status of the object (which is one potential solution to some causality issues) may not be an option as the original agent may no longer be available to control it.

Tools for making and using recordings

Having explored some of the potential requirements of a general underlying record and reuse facility, we now consider some of the tools that we might wish to provide to users so that they can effectively make use of it.

Recording tools

We need tools for making recordings, possibly including:

- requesting that something be recorded or that (assuming that everything is recorded) that parts of a recording are made available to an application.
- specifying what media are to be recorded and ensuring that these are synchronised and the results are named and stored appropriately.
- supporting re-recording, in particular allowing participants to re-enact parts of a experience against an original recording in order to improve them or get them right (like overdubbing in audio and video recording).
- monitoring an ongoing recording.

Analysis tools

We need tools for analysing and tagging recorded data, possibly including:

- Ways of specifying what the user is interested in accessing from a recording including: target objects (specified by attributes); target times (e.g., start and end); target locations; and target events (e.g., being interested in encounters between specific people).
- Generating metadata describing the recordings in an appropriate format (RDF?).
- A suite of (common/standard) tools for statistically analysing user activity and system performance.
- Tools for visualising the contents of recordings and enabling rapid review of material.

Tools for replaying recordings

We may need tools for replaying recordings, including:

- Interfaces to browse lists of recordings, select one and replay it as if it were happening live and view it as an external observer from different viewpoints.
- Replay other recorded media (e.g.. video) alongside a system recording.
- Replay multiple recordings from potentially different times in an interleaved way as if they were all happening live.
- Trigger the replay of recorded data mixed in with live data.
- Specify different relationships between different replays (and live) including temporal relationships (playing forward/backward fast/slow).

Editing, composition and rendering tools

We need to develop a series of tools for being able edit, combine and output recordings, possibly including:

- Tools to cut up recordings, splice them together and otherwise edit them (e.g., change the attributes of the data items).
- Tools for configuring and controlling the relationships between multiple and nested recordings.
- Tools for exporting them in formats that can be imported by animation packages.

Technical challenges and approaches

This section identifies some of the underlying technical challenges that need to be addressed in meeting the requirements and developing the tools outlined previously.

Common structure and formats for recordings

We need to agree a common format for stored recordings that:

- Includes the ability to record system-state.
- Supports both state and event based approaches.
- Supports linking to existing recording formats for other media (audio/video).
- Needs to be linkable to other data and metadata formats.

- Supports jumping to different points within a recording.
- Can be implemented within various platforms (Equip/Elvin).
- Might be realised as a set of cross-indexable recording files (e.g., a data recording and an audio/video recording).
- Resolves issues (e.g. in current version of EQUIP) with recordings presuming particular versions of data types and other components in order to be interpretable. Implies explicit versioning and/or support for migration to new types/versions.

A naming scheme for recordings

We need a naming scheme for our recordings that:

- Specifies dataspace (or other system organisation reference point).
- Specifies time.
- Specifies mediatype (data, video etc).
- Specifies other aspects of the relationship between a particular recording and experience to which it relates, e.g. how it was displayed, where, and how it relates to other recordings
- Can cope with multiple logs from:
 - Different components
 - Different media
 - Related dataspaces
 - Windump, tcopdump, sensor dumps.

Extending distributed data models

We need to extend our data models to include recorded information:

- We may need to extend fundamental dataspace operations to include temporal references:
 - refer to previous event like this one (or nth previous)
 - refer to next event like this one (or nth next)
 - refer to nearest event (before/after) a given time
- We may need to introduce a higher level abstraction such as a ‘temporal link’ that lets an application specify how to access recorded data alongside live Data. This might be an extension to dataspace pattern matching operations and/or standard data item types that describe the relationships between concurrent data spaces (which may also be required to support general scalability).

Replay challenges

Replay interfaces may need to support interpolation between recorded events, both forwards and backwards.

Resolving the issue in Equip with overloading of methods, i.e. that the significance of a particular recorded event or data item may be both in the particular values of its fields as recorded and also in the (invisible) type-specific over-riding of particular methods. For example, each kind of 3D renderable item “knows” – in arbitrary C++

code – how to render itself, and so recreating this rendering requires access to the particular code used at the time in that version/implementation of the recorded item.

Where are multiple recordings related to one another – is this the job of the application (as is currently the case with using multiple Equip dataspaces) or will there be underlying system support (some equivalent to the mechanisms of locales and temporal links in MASSIVE-3).

Transparency – do different components see the difference between recorded and live data?

Correctly filtering the data to be replayed – and coordinating the applications to be running during replay – so that deterministic events are either re-generated by applications or re-played from the recording, but not both.

Similarly, correctly filtering the data to be re-introduced into live events to replay only the sufficient and necessary subset of information that is required to recreate those aspects of the historical event that are intended (taking account of the applications that are now running and their characteristic responses).

Processing and re-mapping information being replayed in a context in which the same thing may be replayed more than once concurrently, or in which the recorded data is still valid in its own right (e.g. in a recursive recording). In other words, data and events that appear similar to those recorded must be introduced, but at least the system must know that they are different, e.g. so that a movement of my “ghost” is not interpreted as the “current me” moving.

Re-purposing or other (potentially) arbitrary remapping and filtering performed during replay, for example some known work on adapting motion-capture data for variant positions and postures of characters when replayed.

Interaction challenges

Do we want to consider the challenges involved in supporting (probably limited) interaction with recorded material? This would involve considering what interactions are possible or how multiple versions of histories can be generated to provide the illusion of interacting with the past.

Disconnection and delay

What additional challenges are raised in situations where we routinely expect disconnection/reconnection between components?

Storage and replication

If recordings become large, we may need to extend our current replication models (which operate at the level of a dataspace?) to operate at some finer level:

- temporal replication that replicates that data within a specific temporal window
- or that replicates the data within a given window around the current point in the recording (e.g., reading a few seconds ahead of the current position)

Efficiency and compression may also need to be considered, e.g. as in MPEG-4.

References

- [1] Ackermann, P., Direct manipulation of temporal structures in a multimedia application framework, *Proceedings of ACM Multimedia (MM'94)*, November 1994, pp. 51-58.

- [2] Barrus, J.W., Waters, R.C. and Anderson, D.B., Locales: Supporting Large Multiuser Virtual Environments, *IEEE Computer Graphics and Applications*, 16(6), November 1996, pp. 50-57.
- [3] Candan, K.S., Prabhakaran, B., Subrahmanian, V.S., CHIMP: A framework for supporting distributed multimedia document authoring and presentation, *Proceedings of ACM Multimedia (MM'96)*, Boston, USA, November 1996, pp. 329-340.
- [4] Frécon, E., Greenhalgh, C. and Stenius, M., The DIVE-BONE – an application-level network architecture for Internet-based CVEs, *Proceedings of ACM Virtual Reality Software and Technology (VRST'99)*, London, UK, December 1999, pp. 58-85.
- [5] Ginsberg, A., Ahuja, S., Automating envisionment of virtual meeting room histories, *Proceedings of ACM Multimedia (MM'95)*, San Francisco, USA, November 1995, pp. 65-75.
- [6] Gliether, M., Retargetting motion to new characters, *Proceedings of ACM Computer Graphics (SIGGRAPH'98)*, Orlando, USA, July 1998, pp. 33-43.
- [7] Greenhalgh, C., Flintham, M., Purbrick, P., Benford, S., Applications of Temporal Links: Recording and Replaying Virtual Environments, *Proceedings of IEEE Virtual Reality VR 2002*, Orlando, Florida, 2002.
- [8] Greenhalgh, C., Purbrick, J., Benford, S., Craven, M., Drozd, A. and Taylor, I., Temporal links: recording and replaying virtual environments, *Proceedings of ACM Multimedia 2000*, L.A. October 2000.
- [9] Greenhalgh, C., Understanding the Network Requirements of Collaborative Virtual Environments, in *Collaborative Virtual Environments: Digital Places and Spaces for Interaction*, Churchill, Snowdon and Munro (eds.), 56-74, Springer-Verlag London Ltd, 2000
- [10] Imai, T., Johnson, A., and DeFanti, T., The virtual mail system, *Proceedings of IEEE Virtual Reality (VR'99)*, Houston, USA, March 1999, p. 78.
- [11] Kanade, T., Rander, P., Vedula, S. and Saito, H., Virtualized reality: digitizing a 3D time varying event as is and in real time, in Ohta, Y. and Tamura, H. (eds.), *Mixed Reality, Merging Real and Virtual Worlds*, Springer-Verlag, 1999, pp. 41-57.
- [12] Leigh, J., Johnson, A., DeFanti, T., Brown, M., et al., A review of tele-immersive applications in the CAVE research network, *Proceedings of IEEE Virtual Reality (VR'99)*, Houston, USA, March 1999, pp. 180-187.
- [13] Luttermann, H. and Grauer, M., VRML History: Storing And Browsing Temporal 3D-Worlds, *Proceedings of ACM Virtual Reality Modeling Language Symposium (VRML'99)*, Paderborn, Germany, February 1999, pp. 153-181.
- [14] MASSIVE-3, <http://www.crg.cs.nott.ac.uk/research/systems/MASSIVE-3>
- [15] Minneman, S.L., Harrison, S.R., Where were we: making and using near-synchronous pre-narrative video, *Proceedings of ACM Multimedia (MM'93)*, August 1993, Anaheim, USA, pp. 207-214.
- [16] Vazirgiannis M., Theodoridis Y. and Sellis T., Spatio-temporal composition and indexing for large multimedia applications, *Multimedia Systems*, 6(4), 1998, pp. 284-298.
- [17] World Wide Web Consortium, Synchronized Multimedia Integration Language (SMIL 2.0), W3C Recommendation, Edited by Jeff Ayers, Dick Bulterman, Aaron Cohen *et al.*, 7th August 2001 (www.w3.org/TR/smil20).