TwinSpace: an Infrastructure for Cross-Reality Team Spaces

Derek F. Reilly, Hafez Rouzati, Andy Wu, Jee Yeon Hwang, Jeremy Brudvik, and W. Keith Edwards
GVU Center, Georgia Institute of Technology
85 5th St. NW Atlanta GA, USA, 30308
{reilly, hafez, andywu, brudvik, jyhwang, keith}@cc.gatech.edu

ABSTRACT
We introduce TwinSpace, a flexible software infrastructure for combining interactive workspaces and collaborative virtual worlds. Its design is grounded in the need to support deep connectivity and flexible mappings between virtual and real spaces to effectively support collaboration. This is achieved through a robust connectivity layer linking heterogeneous collections of physical and virtual devices and services, and a centralized service to manage and control mappings between physical and virtual. In this paper we motivate and present the architecture of TwinSpace, discuss our experiences and lessons learned in building a generic framework for collaborative cross-reality, and illustrate the architecture using two implemented examples that highlight its flexibility and range, and its support for rapid prototyping.

Author Keywords
Cross-reality, collaborative virtual environment, tuplespace, RDF, interactive room, smart room, ontology, virtual world.

ACM Classification Keywords
H5.3. Information interfaces and presentation (e.g., HCI): Group and Organization Interfaces----Computer Supported Cooperative Work.

General Terms
Human Factors.

INTRODUCTION
A common theme of research for a number of years in the ubiquitous computing community has been the creation of 'smart' or 'interactive' spaces to support group work. Systems ranging from the iRoom [1] to the NIST smart room [2] to i-LAND [23], and others have explored the use of specialized 'smart rooms', with a wealth of display devices, sensors, and specialized input devices, to support and enhance collaborative work.

Generally, however, these systems have been designed for collaborative teams in which all the members are collocated in a single smart space. Remote participants have not had access to the wealth of technology in these spaces. Common methods for including remote collaborators—such as a conference call, or video teleconferencing—place remote collaborators at a disadvantage, as they lack both the benefits of collocation with their collaborators, as well as the range of technological supports that are present in the smart space.

In our work we explore the fusion of interactive or smart room technology with virtual worlds (as has been proposed by several authors [8,9]), specifically in support of mixed-presence collaboration. We set out to redress the imbalance between collocated and remote participants by creating 'virtual smart spaces' that remote collaborators can join, and which are deeply interconnected with the physical smart space through pervasive computing technology. These virtual smart spaces provide capabilities similar to—and in some ways even beyond—those provided by their counterpart physical smart spaces. For example, virtual worlds provide effectively perfect 'virtual sensing', meaning that the system has accurate, fine-grained, real-time information about participants' locations and orientations in the virtual space. These spaces also can allow fluid, dynamic configurability and control in ways that may be difficult or impossible in physical smart spaces, which still must operate under the constraints of physics. Perhaps most importantly, combining the sensing and interaction capabilities of both physical and virtual smart spaces may permit collocated groups to actively collaborate with remote persons without forfeiting the practices of collocated collaboration, or the spatiality and interactive affordances of the physical environment.

There are, however, a number of research questions this arrangement raises. First, how do we map the virtual and physical spaces together? Should the virtual space correspond spatially with the physical space, or should mapping focus on shared content, shared activity or some combination of these or other dimensions? Second, how does interconnectivity between physical and virtual spaces provide richer opportunities for collaboration than a conference call does? Third, what are the requirements for adaptability in how physical and virtual spaces are combined? Is a static mapping between physical and virtual required to 'ground' collaboration, or can collocated and remote participants navigate a dynamic range of physical-virtual integrations?

UIST'10, October 3–6, 2010, New York, New York, USA.

In this paper we present TwinSpace, our software infrastructure for connecting interactive workspaces and collaborative virtual worlds. TwinSpace provides a robust platform for rapid prototyping of applications that span physical and virtual spaces, allowing for comparative evaluation of designs; the system is applicable to a wide range of physical settings, virtual environments, and collaborative activities. By facilitating connectivity across a heterogeneous set of displays, sensors and interactive devices in both the physical and virtual worlds, TwinSpace provides a flexible infrastructure for exploring and implementing collaborative cross-reality (CoXR) environments.

Our primary contribution is a novel infrastructure for building CoXR environments that provides the following four key features:

1. A *communications layer* that seamlessly links the event-notification and transactional mechanisms in the virtual space with those in the physical space.
2. A *common model* for both physical and virtual spaces, promoting the interoperability of physical and virtual devices, services and representations.
3. A *mapping* capability, managing how physical and virtual spaces are connected and synchronized.
4. Specialized virtual world *clients* that participate fully in the larger ecology of collaborative cross-reality devices and services.

The remainder of this paper is organized as follows. After a consideration of related work, we present the systems model of collaborative cross-reality supported by TwinSpace. We then present the infrastructure in more detail in terms of challenges faced and lessons learned while implementing a generic framework for collaborative cross-reality. Following this we illustrate the architecture using two implemented examples that highlight its flexibility and range, and its support for rapid prototyping.

RELATED WORK

The TwinSpace architecture is inspired by work in smart and interactive team rooms, collaborative virtual environments and worlds, and collaborative mixed and cross-reality systems. In this section we outline how TwinSpace builds on and departs from earlier research.

Interactive and Smart Workspaces

So-called 'smart' or interactive collaborative workspaces have been a major focus of ubiquitous computing research. Dominant themes in this work include interaction techniques [1,12], sensing and context awareness [2,10] and software/hardware interoperability [1,11]. Each of these themes is also present in the TwinSpace design. Such systems typically support collaboration within one space only, however, and do little to assist remote collaborators. By contrast, TwinSpace provides a broad capacity for defining sensing and interaction behaviour in-room, and extends this capability to include devices and services in the virtual world.

Semantic web technologies have been used in a number of research projects to provide formal reasoning services in collaborative smart spaces [10,13,14,15]. One outcome of this work is that several ontologies dedicated to pervasive and smart space computing have been defined [13,14,15]. TwinSpace extends this approach by permitting reasoning over the combined state of the physical and virtual spaces. More importantly, TwinSpace uses ontology in ways that are unique to CoXR environments—to provide a common model over physical and virtual entities and to determine how physical and virtual spaces are combined.

Collaborative Virtual Environments and Virtual Worlds

Collaborative Virtual Environments (CVEs) provide flexible 3D landscapes within which distributed individuals can interact with each other and with data representations. Typically, these systems create an immersive experience, and employ natural interaction techniques including gesture tracking and haptic feedback. Churchill and Snowdon [16] cite support for "more complex, highly negotiated and interwoven collaborative tasks" as an advantage of a CVE over traditional CSCW applications. CVE work has influenced the TwinSpace design, however TwinSpace emphasizes *mixed presence* (collocated and remote) collaboration that does not mask the real world, but instead makes it a vehicle for working simultaneously with all collaborators. The framework provides base support for 'immersive co-presence' (through support for natural interfaces, the management of shared perspectives, and the representation of collaborators in both spaces), but this is viewed as one among several possible strategies for connecting real and virtual.

Online virtual worlds are increasingly accepted as a way for remote users to 'meet' for collaboration. Typically, the assumption with these systems is that participants are isolated physically from one another, and connect to a shared virtual world server using a personal computer. Second Life, while designed as a primarily social virtual world, has been appropriated for business events, lectures and meetings. OpenWonderland [3], Croquet [4], and commercial products such as OLIVE, Sametime 3D [5] and Qwak Forums provide online virtual environments supporting collaboration over digital media. Typically these systems provide no special affordances for collaborators who happen to be physically collocated. TwinSpace builds on an online virtual world engine to connect these "anytime, anywhere" remote clients with smart physical spaces and the collocated team members inhabiting them.

Mixed Presence and Cross Reality

Mixed presence groupware [17] includes all systems intended to support simultaneous collocated and remote collaboration, while *cross-reality* systems integrate real and virtual worlds in some useful way [8], possibly but not exclusively for collaboration. TwinSpace is the first generic

framework we are aware of for cross-reality, mixed-presence groupware.

The Sentient Computing project [9] was an early cross-reality effort that fused physical sensor networks with virtual worlds by connecting their event models, in a manner similar to TwinSpace. This project had the goal of visualizing sensor activity and not promoting collaboration, however. Recent work by Lifton et al. [8] has explored the same metaphor in support of sharing experiences, media, and informally collaborating, but does not provide a generic architecture for integration with collaborative spaces, and does not explicitly address the needs of collocated groups. A number of other projects have explored elements of collaborative cross-reality, including systems to support mixed presence tourism [19] malleable office spaces [6], and tabletop systems [7]. All of this work considers specific applications or collaborative scenarios, and does not put forward a general architecture. Furthermore, this work has largely considered 'portal' models, which provide standalone interactive device assemblies that connect with the virtual world [6,7]. While a portal's physical location often carries implicit meaning (e.g., this portal is in my office [6]), they are not strongly integrated with their physical surroundings. We believe that collaboration through single portals may reduce the natural spatiality of many kinds of collocated group work by placing focus on a single point of collaboration with remote participants. A generic platform for collaborative cross reality should additionally permit more direct integration with the physical environment, which may be necessary to allow collocated collaboration to remain grounded in the physical space.

SYSTEMS MODEL

TwinSpace provides a generic systems model for CoXR development. At the highest level, TwinSpace connects physical spaces with counterpart virtual spaces so that work in the physical space can remain situated in the physical space without excluding remote collaborators connected to the virtual world. Individual remote users can connect to the virtual world from anywhere using a standard online virtual world client interface, allowing collaboration with collocated groups in one or more physical rooms. The sensor and device infrastructure in physical rooms can be synchronized to the services and virtual objects present in the virtual world interface.

As a generic platform for prototyping and experimentation, the TwinSpace infrastructure permits a range of techniques for integrating physical and virtual spaces. For example, events generated by the smart room infrastructure in the physical space—such as sensor or input device events—can be published and reported in the virtual space. Likewise, events in the virtual space—such as an avatar entering a 'cell' or region, or a new user logging in—can be published in the physical world. Further, the system can define spatial and structural correspondence between the physical and virtual spaces, to support co-presence across the two spaces (a mapping that we call 'twinning' the spaces), and can maintain representations of all parties across both spaces (creating avatars in the virtual space for physical participants, for example, or indicating in the physical world when virtual participants are present in a corresponding region of the virtual space). Spaces may also be connected through functional 'contact points' (for example, a large virtual whiteboard might be paired with a physical interactive whiteboard), whether the real and virtual spaces correspond spatially or not.

The TwinSpace infrastructure allows applications to dynamically reconfigure the integration between the physical and virtual spaces. Specific integrations can be added, altered or removed based on direct commands, user interactions or changes in context. For example, the system can allow applications to remap the devices in the physical room to different regions of a virtual world based on collaborative activity, or an overview camera's perspective may change when a new participant enters an active collaborative region in the virtual world, or the open documents in a virtual space might be replaced when a group switches to a different project.

TwinSpace also has the ability to connect multiple interactive rooms to a virtual space simultaneously, to support multiple, disjoint groups of collocated participants. The rooms can map to separate virtual regions (for example, when connecting offices to a virtual office space [6]), or to the same region (e.g., to allow physically separated creative teams to work together), or they can overlap.

Architecture Overview

The TwinSpace architecture is outlined in Figure 1. TwinSpace combines a virtual world engine and an interactive room infrastructure as balanced parts of a cross reality platform.

On one side of TwinSpace is the OpenWonderland virtual world engine [3]. We chose to build on the open source OpenWonderland platform because it provides rich, programmatic control of the virtual world, the ability to use desktop applications in-world, event-based notification of many in-world events, and well-defined extension points on both the server and the client side [3].

The physical room infrastructure builds on the Event Heap [1], a simple and flexible communications platform that permits the creation of a wide range of room-level interactive capabilities and services [12]. Customized in-room OpenWonderland clients respond to and generate event heap messages[1] alongside other room-based sensors and services.

[1] Clients can also be directly connected to interactive devices using native protocols. We have found this useful when building self-contained client interfaces using devices such as orientation sensors or fiducial trackers.

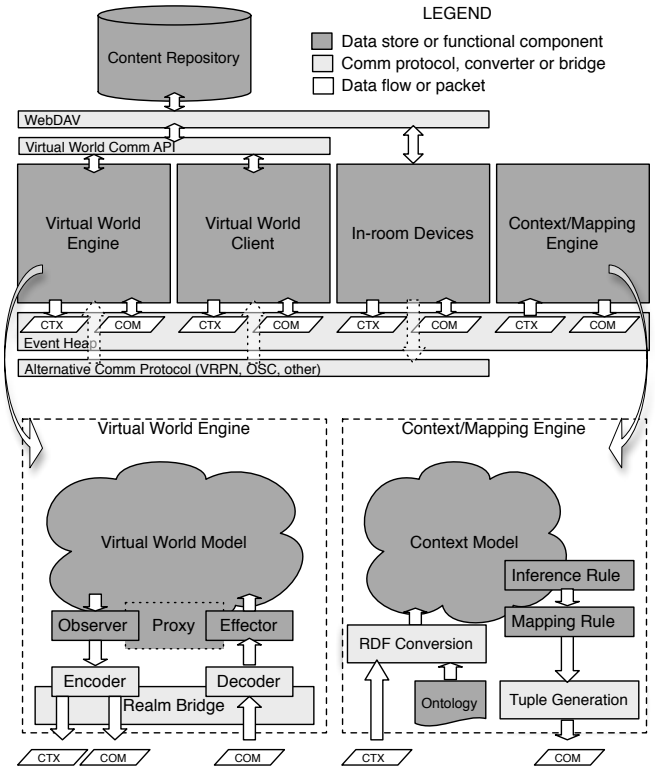

Figure 1. The TwinSpace architecture. Context tuples (CTX) flow from the virtual world, virtual world clients and in-room devices and services into the Context/Mapping Engine, which generates Command tuples (COM). Command tuples can also be generated directly by other components.

Connecting these two sides are custom components providing bi-directional event delivery across spaces, and event translation into formats that are syntactically valid and semantically meaningful for the destination space. A shared model of entities (resources, individuals, events and artifacts) across physical and virtual spaces helps to enforce consistency in message structure and content across realms.

Finally, the TwinSpace infrastructure includes a rule-based translation layer that supports multiple, pluggable methods controlling how entities are represented across realms, and how those representations are combined to give a coherent manifestation of one space in the other.

ARCHITECTURAL CHALLENGES

In this section we discuss the TwinSpace architecture in more detail, calling out architectural challenges and decisions that are unique to collaborative cross-reality environments.

Choosing Client, Server or Both

From an architectural standpoint, we felt it reasonable to define a custom, direct connection between physical and virtual spaces. Events occurring in the virtual world should not be communicated to a connected physical room through an arbitrary graphical client, nor should room-level events not directly controlling a virtual world client require such a client simply to communicate with the virtual world.

This may be at odds with the architecture of online virtual worlds, however, which often employ a client-server model designed around the need to synchronize a 3D model across largely homogeneous but widely distributed clients [24]. While other approaches do exist (e.g., see the peer to peer strategy adopted by OpenCroquet [4]), our experiences using OpenWonderland are valuable as an example of using client-server virtual worlds in CoXR.

Client-server models offer different APIs at each level. This can be a challenge for CoXR systems, which require more direct access to the virtual world model than is typically available on a client, but may still require certain client-side features. For example, OpenWonderland provides rich support for document creation in the client API (such as dynamically mapping a document load request to an appropriate renderer), but very little support on the OpenWonderland server. In addition, clients support multithreading while the server-side transaction and synchronization services severely constrain the creation and management of threads. There are technical limitations in the client API also: for example, the client maintains a subset of the virtual world model (its CellCache), and so can't be used to conduct global searches for documents and other entities without opening specialized comm channels with the server. The OpenWonderland server also maintains an event notification thread intended for server-side components, simplifying in-world event sensing on the server.

After evaluating the use of custom headless clients and combined server+client approaches, TwinSpace connects with the virtual world largely through a custom service called the Realm Bridge (Figure 1), implemented deep in the lower layers of the OpenWonderland server API (to gain control over multithreading and initialization). Its key features are an extensible base communications infrastructure (supporting multiple protocols), support for one-to-many relationships between virtual worlds and physical spaces, and automated syntactic conversion between in-room and in-world event formats. In-world components that use this service reside on the server, and we duplicate essential features from the client API where necessary. In addition, we define a broad set of hooks into the standard clients so they can participate in in-room interactions: in essence, when an interaction in-room controls a camera view onto the virtual world, or controls an avatar that is bound to a client, the corresponding client is used. All other communication between virtual and physical occurs through the bridge.

Synthesizing Physical and Virtual Event Models

One approach in cross-reality systems design is to extend a room-based sensor/actuator model into the virtual world by equating the virtual world's event model with a highly instrumented space. This is complicated by the fact that the kinds of events generated in a virtual world can differ from

those of a physical smart room. In the case of collaborative virtual worlds, certain events carry high semantic value for collaboration (e.g. someone edits a shared document in-world, someone leaves a room), and there may be no reliable way to sense the same level of collaborative event in a physical space. By contrast, the fact that such events are recorded in the virtual world is no guarantee that they can be interpreted in the same manner as if they were occurring in reality—an avatar may leave the room just to get a better perspective on the 3D environment, or may just happen to be facing a specific shared document, etc. Furthermore, there is a balance to strike between achieving real-virtual parity (in event granularity and type) and acceptable performance, when every low-level sensing event or actuation command requires communicating with a remote virtual world server and possibly updating its clients. TwinSpace provides a clear path for developers to manage these differences and tradeoffs using its Observer/Effector paradigm.

Observers and Effectors (Figure 1) are server-side virtual world components that are roughly analogous to sensors and actuators in the physical world. Whereas physical sensors typically provide raw, low-level data, Observers can work at a range of granularities. They may do preliminary processing or reasoning over low-level events (like object movement), they may be combined together (e.g., to sense when an avatar is in close proximity to a document *and* is facing the document), or may simply forward the events produced by the virtual world (e.g. when someone logs into the world). Adding new sensing capacity in the virtual world is usually a simple matter of extending the base Observer and ObservedEvent classes. Adding a new Effector is similarly straightforward.

An Encoder/Decoder hierarchy (Figure 1) encapsulates the generation and parsing of specific protocols and message formats, and they are associated with Observers/Effectors dynamically. For example, a MotionEffector might be associated with a VRPNDecoder for 6dof object manipulation, and an Event Heap TupleDecoder for coarser command-driven operations.

Simplifying and Standardizing Language

Maintaining a consistent message syntax is important for any distributed system. In particular, interactive spaces built using the Event Heap with more than a few components become unwieldy and brittle otherwise. CoXR environments are further challenged here, since the terminology used in-world and aspects of the referent entities themselves can differ from their in-room counterparts. As a result, it can be difficult to address physical and virtual displays in the same way, for example: virtual displays may not need to be turned on or off, or the concept of display might not exist at all.

TwinSpace addresses these issues by weaving an ontology throughout the system, based on the SOUPA [9] ontology for interactive smart spaces. Originally included in TwinSpace to permit context reasoning, the ontology has been most useful as a tool to structure development. The TwinSpace ontology includes SOUPA's Device, Person, Location and Action classes, and adds support for a range of mapping relationships between entities across realms, discussed below.

Using the ontology helps answer the question of how to structure messages passed between devices and services in-room and in-world. It also gives a common model of entities that extends across physical and virtual spaces, facilitating interconnectivity and mapping. However, scaffolding is necessary to make a common model actionable: in the virtual world a device is often a *skeuomorph*, a familiar representation that when clicked on brings up a 2D GUI, whereas in-room a dial controller is a dial controller, for example. Similarly, in-world documents are presented in renderers, while in-room these renderers are situated on displays. We accommodate these and other differences using *proxy* objects in-world.

Proxies are compound entities that combine Observers and Effectors to maintain an abstract model of an ontological element present in the virtual world, like persons, displays, content and rooms. Proxies greatly simplify mapping operations by permitting a shared ontology across physical and virtual spaces, and by encapsulating low-level virtual world commands.

For example, the Display proxy provides a virtual counterpart to physical displays. Displays encapsulate the creation and configuration of in-world document renderers. If a request arrives to place a new type of document on the display, the Display Effector destroys the current renderer, replacing it with an app that can render the document, then loads the document. When content is changed on the Display, a Display Observer communicates this back out to registered rooms. On physical displays we provide a basic listener that can open documents in native renderers. This makes it straightforward to synchronize virtual displays with physical displays that present content using native renderers.

Achieving Dynamic Configurability

Another purpose of the shared ontology is to permit context-driven, rule-based triggers in the TwinSpace framework. We wanted this for two reasons. First, (as discussed) virtual worlds have the equivalent of a very robust sensing infrastructure, limited mainly by what the virtual world engine chooses to make available as state and events, offering interesting possibilities for context-driven integration of physical and virtual. Second, rule-based triggers allow centralized control over reconfiguring spaces (or services in them) and the relationship between spaces via pluggable mapping rules. This focus on mapping is a unique feature of the Context/Mapping Engine (CME, Figure 1). Together with the shared physical/virtual ontology, the CME facilitates the definition of cross-reality configurations ranging from the very straightforward to very complex.

In practice, we have used the context reasoning capacity very little. Our *inference rules* (Figure 1) have been very

basic tests for presence/absence of specific contextual data, aside from one proof of concept experiment that tracked speaking avatars based on their seniority level in an organizational hierarchy. This is partly because much of the context posted from the virtual world is already at a higher-level (e.g., someone has entered/left a room, started to edit a specific document, etc.), and because our use of sensing technologies in-room to date has largely been in support of explicit interactions. We anticipate that context reasoning may become important when attempting to infer activity across realms. For example, identifying who in the physical and virtual spaces are attending to a specific document might require combining information about presence, physical orientation, and recent history.

Mapping rules, by contrast, have been a critical part of our prototyping work. Mapping rules enforce correspondences between entities (such as displays) in either or both physical and virtual spaces, and determine how contextual events are manifested. Mapping rules generate and post Command tuples back onto the Event Heap, causing mappings to be realized in both physical and virtual spaces. Over time we refined the mapping rule design to use indirection and have recently added an ontological *Relationship* element that can be reasoned over, as discussed below.

Mapping rules are defined abstractly in the CME, referring to devices indirectly by *mapping type*. Rules are concretized at evaluation, and a device's mapping type(s) can be altered dynamically. This is done for two reasons. First, the availability of resources in physical and virtual environments can change over time. Second, abstract mappings permit the support of multiple configurations of space and resources (for example, to promote collaboration across multiple physical teamrooms). For example, a single mapping rule involving *primary* and *secondary* mapping types can be used to provide a detail+context presentation of a shared virtual whiteboard in a teamroom with two displays (one labeled 'primary' and the other 'secondary'), and to simultaneously generate a split-screen view on a single display in another room (if the display has both mapping types assigned to it).

In addition to mapping types, entities may hold one or more *relationships* with other entities. For example, an avatar can be twinned with an in-room participant by defining two Person entities that hold a one-to-one *twin* relationship with each other. A display that focuses on a moving or stationary object holds a *track* relationship with it. Arrays of physical and/or virtual displays have the *array* mapping type plus a *group* relationship that specifies ordering. While this work is still preliminary, we feel that it, by allowing relationships to be represented and reasoned over, represents an evolution of context reasoning that is appropriate for cross-reality systems like TwinSpace.

Because they are expressed in RDF, inference and mapping rules can be individually modified, linked together, or swapped out of the CME with minimal system configuration. This is to support experimentation and dynamic reconfiguration. In our prototyping work, when someone logs into the virtual world, different mapping rules can be applied so that a table lights up in-room, or an in-room shared client jumps to the login area, or both. Our speaker tracking proof of concept demonstrates the same for inference rules: the org chart rule described above can be replaced by one that simply specifies that whoever started speaking most recently should be tracked.

Altering the Interaction Model for Group Use

CoXR environments can employ a wide range of perspectives or views onto the virtual world: a view that corresponds to the physical location and orientation of the viewer (in immersive approaches), a 'best-view' close-up of an in-world 2D document, a top-down view, an overview perspective, views that change based on events in-room or in-world, etc. However, CVW systems often build-in assumptions that conflict with cross-reality collaboration: for example, in OpenWonderland clients are bound to an avatar and interaction is designed for a single user via keyboard+mouse.

As a result, we have modified the basic OpenWonderland CVW client to support group use. Most importantly, clients are addressable via command tuples and we decouple the relationship between avatars and clients. These are basic requirements for room-level interactivity, cross-reality mappings, and event-based updates. We provide a core set of services on the client that can be accessed using command tuples: control over the client's configuration, emulation of input events, the ability to locate and track in-world entities, direct control of camera and/or avatar movement using relative or absolute transformations, and a set of camera effects, including basic cut-away transitions, smooth transitions, zooming, and presentation of multiple views.

CASE STUDIES

In this section we discuss our experiences using TwinSpace in terms of two implemented prototypes.

Activity Mappings

This case study describes a CoXR prototype designed to utilize the spatial layout and 'collaboration affordances' of a specific team room (the 'inSpace' lab [25]), in order to promote mixed (collocated and remote) collaboration. More detail about this case study is provided in the video figure.

The team room is composed of three main regions (see Figure 2). The *assemble/array* region supports extended group discussions and sharing content on wall displays. The *aside table* region supports composing, modifying, and discussing content over a tabletop display, and the *aside wall* region supports generating new content on a whiteboard.

We outline two high-level activities the prototype currently supports: creative 'brainstorming' and giving presentations. **Brainstorm** mode configures the three regions of the room to support idea generation and group decision-making.

Figure 2. Three collaboration regions in the inSpace team room and the connected virtual space in *brainstorm* mode. *Aside wall* (A) centers around a large interactive whiteboard. *Assemble/array* (B) uses three wall displays configured around a sensing table. *Aside table* (C) combines an interactive tabletop and vertical display. Remote participants inhabit the virtual space and communicate with co-located collaborators using spatialised audio and telepointers.

Present mode provides facilities for in-room or remote users to give a presentation or guide a discussion with other participants.

Modes are selected using a small touchscreen interface (see Figure 3) attached to one end of the 'assemble' table. When a mode is selected, the CME responds by firing a set of mapping rules, reconfiguring the team room's digital devices as described below.

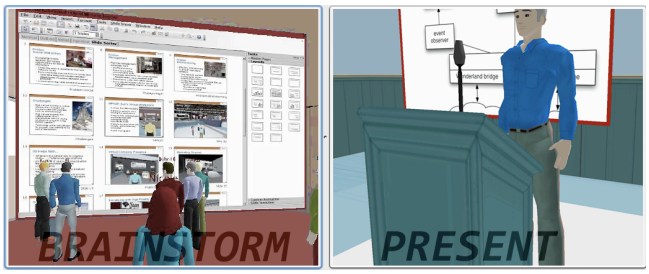

Figure 3. Touchscreen buttons for the Brainstorm and Present modes in the team room.

In **brainstorm** mode, the spatial layout of the room is echoed in-world. Each of the three collaboration regions is mapped to its corresponding region in-world (see Figure 2D). Further, within each collaboration region, displays are mapped to resources that have the same relative position and orientation in-world as the display does in-room.

The wall displays in the *assemble/array* space operate as a unit in brainstorm mode (their CME representations share a *group* relationship and an *array* mapping type). An in-room dial controller (see video figure) provides a way to iterate through sets of in-world documents (more precisely, a set of virtual displays[2]). The wall displays operate as a window onto a subset of the documents in the set. Turning the dial iterates forward or backward along the current set of documents, by giving each wall display client the id of an in-world object (in this case, virtual display) to move to. Pushing and turning the dial will rotate the view of all three displays simultaneously (allowing exploration of the virtual space surrounding each document), snapping back to the default view once the dial is released. Remote collaborators can connect to the dial session by clicking on a virtual dial object that is twinned with the physical control. Their client then presents the same subset of documents visible on the in-room displays on a heads-up display (HUD) layer. A GUI provides complementary controls so that in-world collaborators can also control the session.

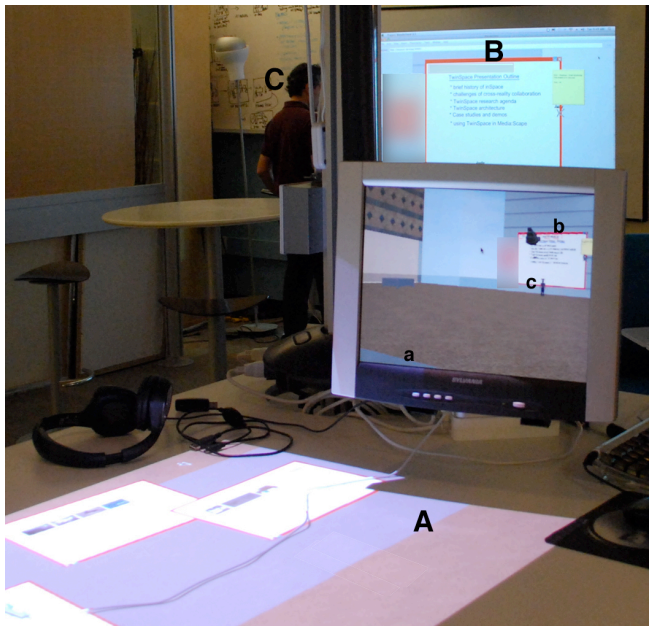

Figure 4. Correspondences in Brainstorm mode. A/a: displays in the *aside table* region show horizontal and vertical perspectives corresponding to the region's in-world location. B/b: the vertical perspective highlights the spatial correspondence between the *aside table* region and the other two regions. C/c: an in-world avatar represents a remote person collaborating with the person at the whiteboard in the *aside wall* region.

In brainstorm mode the *aside wall* region promotes an open exchange among collaborators, regardless of mode of input and physical location. In-room participants can use a pen to draw on the physical whiteboard, and thereby onto the shared document in the virtual space (Figure 2(A)). Remote participants use standard virtual world clients, and can modify the document using mouse and keyboard. On a cocktail table near the whiteboard we provide a tablet computer rendering the document in a native web browser.

[2] The virtual displays are grouped using the same mechanism (group, array) as the physical displays. Selecting display sets is made possible through the *group* relationship.

This is an example of how native renderers can be used to 'twin' virtual and physical displays in TwinSpace.

The *aside table* region permits collocated and remote collaborators to discuss or work on content on a tabletop display. In the region we combine a basic interactive tabletop display with a vertical 'orienting' display on an adjustable boom. On the tabletop we present a top-down view of a small region in the virtual world, and place virtual documents horizontally in this region. The tabletop tracks multiple pens, which can be used to arrange or edit these documents. Remote participants can choose to use a top-down or oblique view over the same region, and can edit the documents using the standard CVW client controls. The vertical orienting display can be positioned in space and rotated about the boom's center. A 3D orientation sensor causes the camera view to update based on yaw and linear acceleration. Consequently, the vertical display allows collaborators at the physical table to peer into the region of the CVW surrounding the area projected on the table (see Figures 2,4). Since the sensor is tightly coupled to the single client, we connect the sensor directly using its native API.

P**resent** mode supports giving presentations to a mixed in-room and remote audience (see video figure). In this mode the display array in the *assemble/array* region is used, and the displays in the two aside regions are dimmed. The three displays are assigned specific mapping types for this mode in the CMV: the left is an *overview* display, the center is the *main* display and the right is the *secondary* display. The overview presents a raised oblique overview of the CVW presentation space, showing the virtual podium, the audience area and the location of presentation content. The main display shows the main presentation content, and the secondary display shows supporting content like video, whiteboard or websites.

The remote audience sees the main content presented above the speaker's avatar, and supporting content to the right. Whether collocated or remote, the presenter logs into the virtual world using a standard CVW client, and identifies themselves as *presenter* by clicking on the podium and selecting 'present' on a popup dialog. This sets their avatar in front of the podium and brings up controls for presentation content.

To support dialog during or after the presentation, the left in-room display is also assigned a *speaker tracking* role in the CMV. When someone other than the presenter speaks in-world, the view snaps to a front-view of that person's avatar. This behaviour is triggered by the CMV based on updates from our Speech Observer. After a few seconds elapse with no speaking, the display falls back to its default 'overview' perspective.

Present mode highlights TwinSpace's ability to fluidly map functionality to devices (in this case to displays). In a different physical environment, each of the *main, secondary, overview,* and *speaker tracking* roles can be assigned to different displays as available and appropriate for that space.

We have been periodically evaluating this prototype during weekly meetings with our collaborators in industry. We have successfully used each of the three regions during brainstorming, however there are some key areas for improvement that we have identified, all related to supporting awareness. First, the in-room collaborators exhibited a need to maintain awareness of who is in a given region in-world. The orienting display provides some indication of this, but is only located in one of the three regions. Second, the remote collaborators lamented that there was no virtual representation of their in-room counterparts. Finally, there was limited awareness of ownership of action: when applications are 'taken control of' in-world, there is no indication on the application of who is interacting with it. Similarly, when one of the in-room participants takes control of an application (for example, the virtual whiteboard in the aside wall region), there is no indication of who in-room is interacting with the whiteboard.

Cross-Reality Collaborative Game

Or de l'Acadie is a prototype collaborative game built using TwinSpace. We outline it here to give a sense of the flexibility of the TwinSpace architecture. Set in an abandoned 18th Century French colonial fort, two pairs (French and British sides) compete for valuables distributed throughout the town, with the goal to collect the most valuables within a time limit. The game emphasizes asymmetry in both game controls and team dynamics. Each British player controls a single soldier using keyboard controls. Players are not collocated, and can communicate with each other only when their soldiers are within earshot. Each soldier is equipped with a static top-down map of the fort, and can collect a limited number of riches before needing to return to a location at the edge of the fort to stash them. The French side consists of a Mi'kmaq Indian mystic who is able to communicate with spirits, and a deceased French *gendarme*. The French players are physically collocated, and use a range of physical controls to compete for the valuables. While the 'living' players (French and British) are constrained by the laws of physics, the Gendarme moves freely through walls and in the air, but is unable to collect valuables. Instead, he communicates with the mystic who collects them. The mystic pushes a cart, and so can collect more riches than the soldiers, but as a result he cannot move quickly, or climb stairs. Luckily, the Gendarme can transport the mystic to any location in the fort by creating a portal.

The gendarme is equipped with three simultaneous views: a top-down view of the entire fort, a wide-vista first-person view distributed across three screens, and a third-person view of the mystic's current location. The player controls the gendarme's first-person view using a SpaceNavigator 3D mouse. The table always displays a top-down view of the entire fort, but its front clipping plane corresponds with the gendarme's current altitude. When the gendarme locates valuables he can create a portal between where the mystic

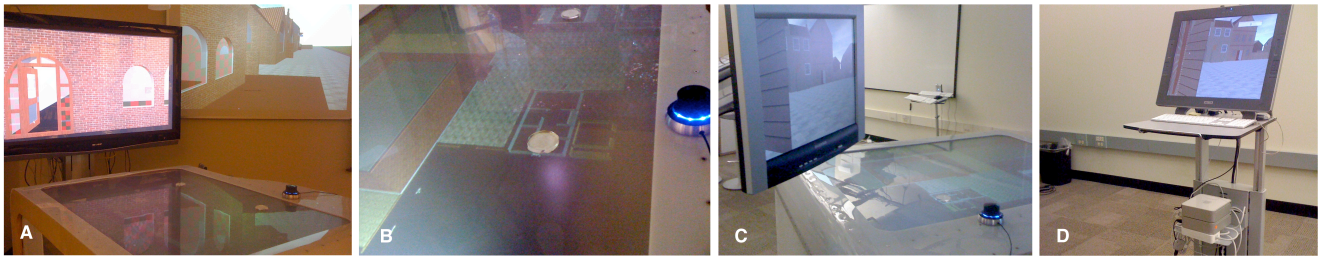

Figure 5. Interfaces of the French team in Or de l'Acadie. (A) the gendarme's first person view is presented on three large wall displays (two shown here). (B) in addition, a top-down view of the fort is used to place portal endpoints using fiducial markers. The 3D mouse (on right) is used to simultaneously move the first person perspective and adjust the clipping plane of the top-down perspective. (C) the gendarme can also see the viewpoint of his teammate on a boom display (on left). (D) the mystic's interface. This player pushes the physical cart around on the carpet to move her avatar, returning to a marked location on the floor to walk through a portal. Items are selected using the touch display.

currently is and the location of the valuables. Portals are created by placing tangible objects at the two endpoints of the portal on the table's top-down view of the fort. The mystic walks through the portal to enter the next location. Finally, the gendarme has a third person view of the mystic on a display on a boom. The view follows the participant, and rotating the boom changes the viewpoint relative to the mystic's orientation. In this way the gendarme can help guide the mystic to the valuables in a location if he is having difficulty.

The mystic and his cart are controlled using an assembly we call the TwinSpace Trolley. A 24" touchscreen is placed on a physical trolley, providing the first-person perspective of the mystic. The trolley is maneuvered around the physical space to navigate a mapped virtual space. Valuables are placed in the cart by touching them on the display. The trolley's position is determined using a sensor assembly involving an RFID reader (reading tags in the carpet), a 3D orientation sensor and an optical mouse ('reading' the carpet). This sensor data is combined by a small program that transmits position and orientation updates to the event heap. The physical room area is mapped to the virtual space in the CME by calling custom Java code (in this case a basic shift and scale is applied). The resulting command to change position is sent to the trolley's virtual world client. When the mystic walks through a portal, an observer sends notification to the CME, which updates the shift values accordingly.

The gendarme's main control is the SpaceNavigator 3D mouse. An in-room application receives data from the mouse via the OSC protocol, and converts this into Event Heap command tuples containing distinct transforms for each vertical display CVW client, and into a clipping plane value for the tabletop CVW client. The boom display uses the same configuration as in the previous case study, except that its CVW client also receives the trolley's position updates from the CME. Updates to the position and orientation of the tangible portal controls are converted from TUIO protocol messages into Event Heap context tuples by a custom application. The CME captures these and sends updates to the twinned portal objects in the virtual world, via a MovementEffector.

CONCLUSION

We have presented TwinSpace, an infrastructure for cross-reality collaborative spaces. TwinSpace is designed to support deep interconnectivity and flexible mappings between virtual and physical spaces—two basic capabilities that facilitate cross-reality application prototyping and evaluation.

The requirements of mixed presence collaboration are not met by standard online virtual worlds, nor by traditional smart room infrastructures. We have illustrated, however, that despite architectural differences between the two models, a coherent architecture for CoXR can be defined using these very different technologies.

By promoting common abstractions in physical and virtual spaces, by streamlining event delivery across spaces, and through its sophisticated capacity to specify and trigger mappings of large and small scale, TwinSpace is a flexible framework for prototyping, evaluating and configuring interactive cross-reality environments.

ACKNOWLEDGMENTS

This work was funded by NSF grant IIS-0705569, and a research collaboration grant from Steelcase Inc. We also thank the former Sun Microsystems Labs Project Wonderland team for in-kind and technical support.

REFERENCES

1. Johanson, B., Fox, A., Winograd, T. (2002) The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Per. Comput., 1 (2), 67–74.
2. Stanford, V., Garofolo, J., Galibert, O., Michel, M., Laprun, C. (2003) The NIST Smart Space and Meeting Room Projects: Signals, Acquisition Annotation, and Metrics. Proceedings of ICASSP '03, 736–739.
3. The OpenWonderland Project, http://www.openwonderland.org, retrieved March 2010.
4. Smith, D. A., Kay, A., Raab, A., and Reed, D. P. (2003) Croquet – a collaboration system architecture. Proceedings of the Intl. Conference on Creating, Connecting and Collaborating through Computing

5. Virtual Collaboration for Lotus Sametime. Retrieved Sept.'09. www-01.ibm.com/software/lotus/services/vc4sametime.html
6. Schnadelbach, H., Penn, A., Benford, S., Koleva, B. and Rodden, T. (2006) Moving Office: Inhabiting a Dynamic Building, Proceedings of CSCW '06.
7. Regenbrecht, H., Haller, M., Hauber, J., and Billinghurst, M. (2006) Carpeno: interfacing remote collaborative virtual environments with table-top interaction. Virtual Real. 10(2), 95–107.
8. Lifton, J., Laibowitz, M., Harry, D., Gong, N-W., Mittal, M., and Paradiso, J. A. (2009) Metaphor and Manifestation: Cross-Reality with Ubiquitous Sensor/Actuator Networks, IEEE Pervasive Computing, 8 (3), 24-33.
9. Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A., and Hopper, A. 2001. Implementing a Sentient Computing System. Computer 34, 8 (Aug. 2001), 50-56.
10. Ye, J., Coyle, L., Dobson, S., and Nixon, P. (2007) Ontology-based models in pervasive computing systems. Knowl. Eng. Rev. 22 (4), 315–347.
11. Blackstock, M., Lea, R., and Krasic, C. (2008) Evaluation and analysis of a common model for ubiquitous systems interoperability. Proceedings of Pervasive '08, 180–196.
12. Ponnekanti, S. R., Johanson, B., Kiciman, E., and Fox, A. (2003) Portability, Extensibility and Robustness in iROS. In Proceedings of PerCom'03, 11.
13. Chen, H., Finin, T., and Joshi, A. (2003) An ontology for context-aware pervasive computing environments. Knowl. Eng. Rev. 18 (3), 197-207.
14. Ranganathan, A. and Campbell, R. H. (2003) A middleware for context-aware agents in ubiquitous computing environments. Proceedings of the ACM/IFIP /USENIX Intl. Conference on Middleware '03,143-161.
15. Wang, X. H., Zhang, D. Q., Gu, T., and Pung, H. K. (2004) Ontology Based Context Modeling and Reasoning using OWL. In Proceedings of PerCom'04 Workshops Washington, DC, 18.
16. Churchill, E. F., and Snowdon, D. (1998) Collaborative Virtual Environments: An Introductory Review of Issues and Systems. Virtual Reality: Research, Development and Applications 3(1), pp. 3-15.
17. Tang, A., Boyle, M., and Greenberg, S. (2004) Display and presence disparity in Mixed Presence Groupware. Proceedings of OzCHI'04, 73-82.
18. Milgram, P. and Kishino, F. (1994) A Taxonomy of Mixed Reality Visual Displays. IEICE Transactions on Information Systems, Vol E77-D, No.12.
19. MacColl, I., Millard, D., Randell, C., and Steed, A. (2002) Shared Visiting in EQUATOR City. Proceedings of CVE '02, Bonn, Germany.
20. Johansen, R., Sibbet, D., Benson, S., Martin, A., Mittman, R. & Saffo, P. (1991) Leading Business Teams. Addison-Wesley.
21. Yankelovich, N., Simpson, N., Kaplan, J., and Provino, J. (2007) Porta-person: telepresence for the connected conference room. In CHI '07 Extended Abstracts, 2789-2794.
22. Jena, a semantic web framework for Java. Retrieved Sept. '09. http://jena.sourceforge.net/
23. Streitz, N. A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., and Steinmetz, R. (1999) i-LAND: an interactive landscape for creativity and innovation. Proceedings of CHI '99, 120-127.
24. Sandeep, S., and Zyda, M. Networked Virtual Environments: Design and Implementation. Addison Wesley, 1999.
25. Reilly, D., Voida, S., McKeon, M., Le Dantec, C., Edwards, W. K., Mynatt, E. and Mazalek, A. Space Matters: Physical-Digital and Physical-Virtual Co-Design in the Inspace Project. IEEE Pervasive Computing, to appear.