IST BASIC RESEARCH PROJECT
SHARED COST RTD PROJECT
THEME: FET DISAPPEARING COMPUTER
COMMISSION OF THE EUROPEAN COMMUNITIES
DIRECTORATE GENERAL INFSO
PROJECT OFFICER: THOMAS SKORDAS

Global Smart Spaces

Final Examples of Interaction Techniques Using Multiple Interaction Surfaces

D20

10/01/2004, UNIV. JOSEPH FOURIER/WP7/VERSION 1.0

J. COUTAZ, C. LACHENAL, N. BARRALON, G. REY

IST Project Number	IST-2000-26070	**Acronym**	GLOSS
Full title	Global Smart Spaces		
EU Project officer	Thomas Skordas		

Deliverable	**Number**	D20	**Name**	**Final Examples of Interaction Techniques Using Multiple Interaction Surfaces**			
Task	**Number**	T	**Name**	n/a			
Work Package	**Number**	WP7	**Name**	Interaction Techniques			
Date of delivery	**Contractual**		Dec. 31rst 2003	**Actual**		Jan. 10, 2004	
Code name	n/a			**Version** 1.0 draft ♦ final ⌋			
Nature	Prototype ⌋ Report ⌋ Specification ♦ Tool ♦ Other:						
Distribution Type	Public ⌋ Restricted ♦ to:						
Authors (Partner)	J. Coutaz, C. Lachenal, N. Barralon, G. Rey						
Contact Person	J. Coutaz						
	Email	Joelle.coutaz@imag.fr	**Phone**	+33 4 76 51 48 54	**Fax**	+33 4 76 44 66 75	
Abstract (for dissemination)	This document includes 1) a revision of the ontology for multi-surface interaction proposed in D19, 2) a brief reminder of the technical principles of I-AM, a software infrastructure that implements the ontology, and 3) the description of interaction techniques built on top of I-AM. These techniques illustrate the generality of I-AM in terms of the types of resources composition it is able to support. We conclude the work developed on multi-surface interaction with the analysis of opened UI-centered problems for future research in ubiquitous computing.						
Keywords	Multi-surface interaction, distributed user interface, migratory user interface, Interaction Abstract Machine, coupling interaction resources, mapping digital and physical spaces.						

Table of Content

1 INTRODUCTION

In the initial work plan of WP7, this document entitled "*Final* Examples of Interaction Techniques Using Multiple Interaction Surfaces" was intended to complement or replace Deliverable D18 "*Initial* Design of Interaction Techniques Using Multiple Interaction Surfaces" delivered in October 2003 for the Ivrea review. Based on the recommendations of the reviewers proposed in November 2003, as well as on additional work performed during the last two months of the project, this document now covers three areas:

. Revision to D19 (Final Reference Framework for Interaction Surfaces) delivered in October 2003. D19 includes 1) the description of an ontology that makes explicit the concepts of multi-surface interaction and 2) the description of I-AM[1], a software infrastructure that implements the ontology. I-AM was demonstrated in November 2003 in Ivrea at the final review meeting. Since the writing of D19, the foundations for multi-surface interaction are still valid, but the UML description of the ontology has been modified to improve the generality of the model and consequently to improve our analysis of the concept of multi-surface interaction for future research. This revision was not expected in October 2003.

. Description of the interaction techniques built on top of I-AM as available at the end of the project (December 2003). This section is an update to D18 and includes complementary examples.

. Presentation of user-centred issues for future research that our ontology on multi-surface interaction has permitted to elicit. Some of these issues have been reported in the early phase of the project (Cf. D17). This section, which extends our initial analysis, is another update to D18.

The document is structured according to the three areas introduced above.

. In the next two sections, we briefly present our new version of the ontology for multi-surface interaction (Section 2) and we recall the technical principles of I-AM (Section 3).

. In Section 4, we describe the interaction techniques developed with I-AM. These techniques illustrate the generality of I-AM in terms of the types of resources composition it is able to support.

. In Section 5, we concentrate on the analysis of opened UI-centered problems for future research in ubiquitous computing.

[1] I-AM stands for Interaction Abstract Machine.

2 ONTOLOGY FOR MULTI-SURFACE INTERACTION: THE NEW VERSION

Figure 2.1 recalls the point of departure of our vision for future forms of interaction in Global Smart Spaces. It reads as an extension of the Model of the Human Processor [Card 83] where the "cognition-actuator-sensor" structure of a human actor has its counterpart in the artificial world, and where the artificial and human actors mediate through interaction resources: surfaces and instruments. An actor has sensors and actuators to respectively observe and act on interaction resources. As an *instrument*, an interaction resource mediates the actions that an actor intends to direct to another actor. As a *surface*, an interaction resource provides an actor with a means for making its state observable to another actor.

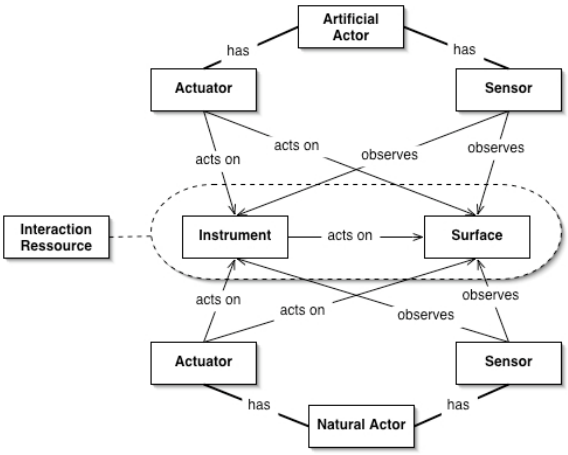


Figure 2.1. The Human Actor and the Artificial Actor mediate through interaction resources.

Figure 2.2 shows the new UML diagram that describes our ontology for multi-surface interaction. The previous UML diagram can be found in Annex 1. As shown in the diagram, natural and artificial actors are a kind (i.e., a subclass) of actor. In turn, an actor is a physical entity. It has sensors and actuators (sort of physical entities) that an actor relies on to respectively observe and act on physical entities. Through observing, acting, and thinking, an actor is able to elaborate information[2]. In turn, information is represented by interactors whose role will be discussed later on in this section.

A physical entity is characterized by physical attributes such as shape, weight, colour. It can act on and/or observe other physical entities. For example, a marble that bumps into another one, acts on the second marble. A physical entity is an interaction resource if an actor has associated an *interaction role* to it. This role can be that of a surface, of an instrument, or both. In addition, a surface is an *action surface* for actor A if A can act on it. Symmetrically, a surface is an *observation surface* for A if it can be observed by A. An instrument serves as a *pointing instrument* for A if it allows A to denote a point on a surface. It is a *text instrument* for A if it allows A to deal with

[2] The notion of information can be refined into conceptual models of the world (and of the self), not shown in the diagram.

character strings. Clearly, the ontology may be extended to cover other types of instruments.

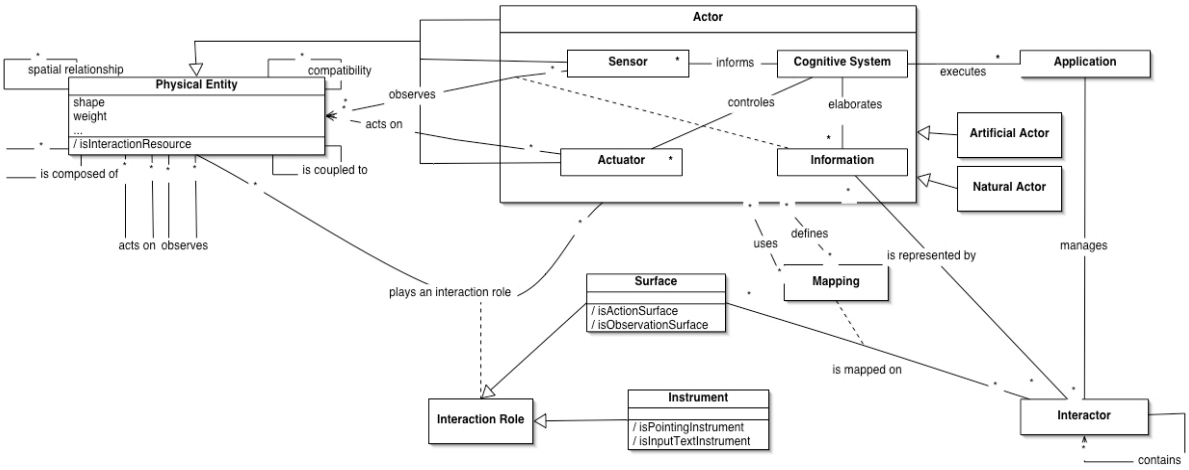


Figure 2.2. Ontology for multi-surface interaction: the new UML diagram.

As an illustration, consider our Magic Table [Bérard 03] shown in Figure 2.3. Here, the artificial and the human actors have assigned the role of surface to the table and two roles (that of instrument and surface) to the physical blue-colored plastic tokens. The camera sensor of the system observes the tokens to compute their location on the table whereas users act on them using their fingers as actuators. For the system and the users, the tokens play the role of instruments to explore digital information. Because they denote location of points on a surface (i.e., the table), they are pointing instruments. In addition, due to their shape (they are flat and large enough), the system and the users can exploit them as surfaces as well: the system expresses a portion of its internal state to the users by projecting digital information on the tokens with a video-projector. (Note that the video-projector is an actuator of the system.) In turn, users can observe the displayed information using their eyes as sensors. As surfaces, the tokens are action surfaces for the system and observation surfaces for the user. But, by observing the physical blue-colored round patches on the table, the system can locate the tokens and assign them the role of instruments.

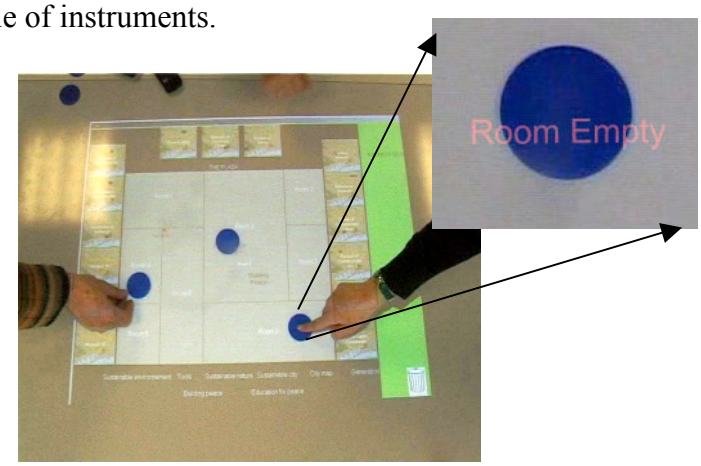


Figure 2.3. The Magic Table where physical tokens play the roles of: instrument and surface.

Coming back to the UML diagram of our ontology, physical entities may be composed of physical entities. For example, an actor is composed of sensors and actuators. Physical entities, actors and their parts, interaction resources and other

physical entities in the world, are related by spatial geometric relationships and coupling.

Coupling is the act of linking two entities so that they can operate together to provide a new set of functions. As shown in the UML diagram, coupling may be performed between entities that have the same role or between entities that have different roles. Figure 2.4 shows an example where users have coupled two instruments. Once they are coupled, the two tokens provide the functions select, rotate, move, and resize digital information. Figure 4.4 shows examples of screens that have been coupled to provide another type of function: "enlarge display area".

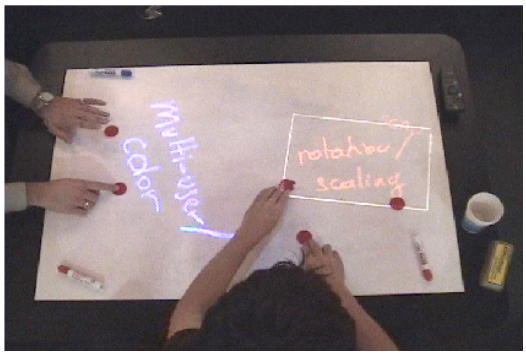


Figure 2.4. Another setting of the Magic Table where tokens are coupled to select and manipulate digital information.

As shown in Figure 2.2, artificial actors run applications whose execution results in the elaboration of new information. In order for actors to share a common understanding of the situation, portions of this information must be represented (cf. Norman's Theory of Action). In our ontology, interactors are information representatives. More precisely, information produced by the execution of an application is represented by the interactors managed by this application. Interactors are made observable on a surface by the way of a mapping function.

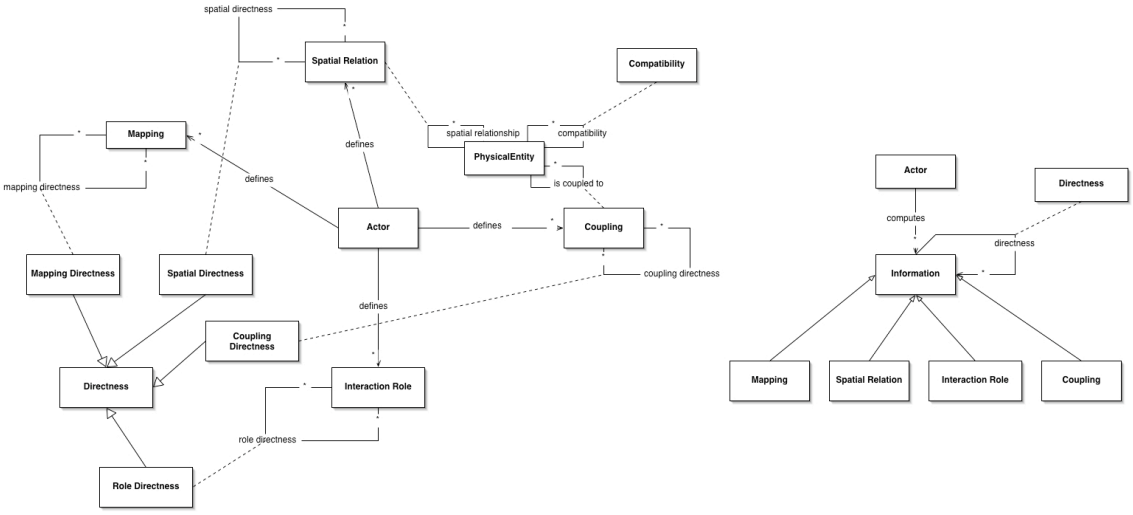


Figure 2.5. Actors define the relationships between the physical entities and between the digital information and the interaction resources.

A *mapping* is a transformation that takes a set of interactors as input and produces an observable tangible rendering on a set of surfaces. Similarly to information, interaction

resources, whether they are used as surfaces or instruments, have states that are represented with interactors. Typically, a mouse used as a pointing instrument is represented by a graphical shape mapped on a screen surface at the point denoted by the mouse position in the physical space.

As shown in Figure 2.5, all of the relations of the ontology (i.e., coupling, mapping, spatial relationships, interaction roles) are defined by actors. Referring to the principles enunciated by Norman in his Theory of Action [Norman 86], we need to consider the directness of these relations, as well as the compatibility between what is established by the artificial actor and what is established by the natural actor. As a simple illustration, let us consider the traditional GUI configuration shown in Figure 2.6.

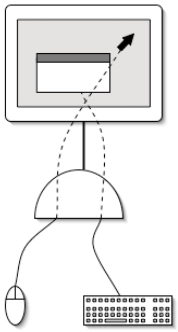


Figure 2.6. Traditional configuration in the GUI paradigm.

Figure 2.7 represents Figure 2.6 configuration in terms of the multi-surface ontology. John and the computer are respectively the natural and the artificial actors. A keyboard, a mouse and a screen are the physical entities of interest. John elaborates a mental representation of the keyboard, of the mouse and of the screen modeled in the UML diagram as the *johnKeyboard*, *johnMouse*, and *johnSurface* entities. Symmetrically, *computerKeyboard*, *computerMouse* and *computerSurface* denote the entities that the computer actor handles along with their roles as instruments and surface. The diagram makes explicit the distinction between the entities and the relationships that the two actors elaborate for the same physical things of the real world.

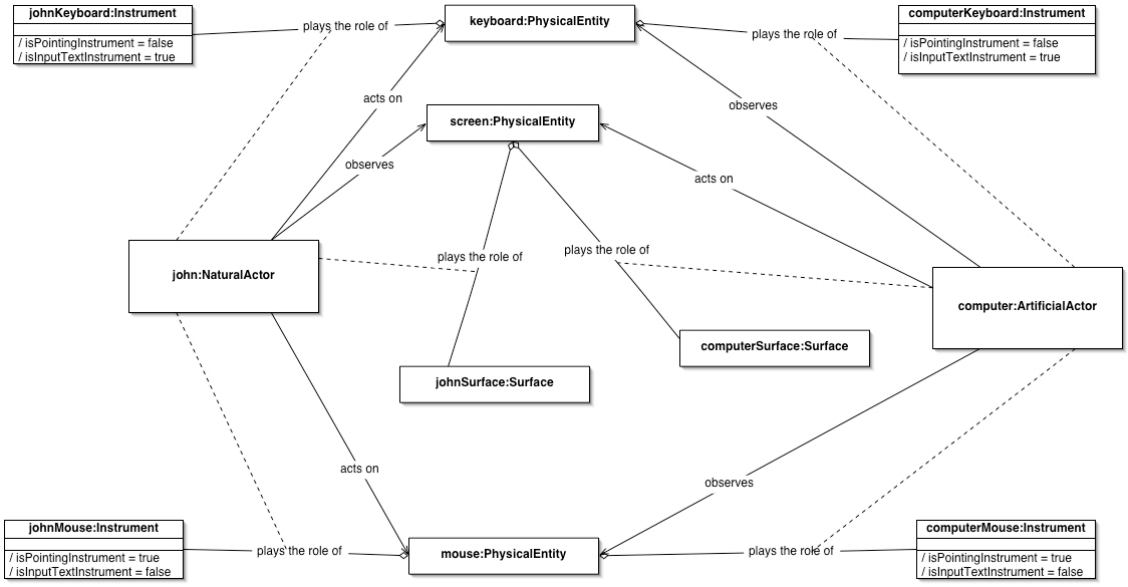


Figure 2.7. The representation of a traditional GUI configuration using the multi-surface interaction ontology.

As a designer, one needs to ask whether *johnKeyboard* is compatible with *computerKeyboard*, and whether there is a direct match between them. Because John is familiar with GUI settings or because of the perceived affordances [Norman 99] of the physical entities (screen, mouse and keyboard), it is reasonable to assert that John has assigned the role of instrument to the keyboard and to the mouse, and the role of surface to the screen. Therefore, role assignments performed by the natural and the artifical actors are compatible and they are direct (the diagram shows a direct link between the physical entities and their representation). However, mismatches may occur at the attribute and property level. For example, when pressing the "A" key of an AZERTY keyboard, the user intends to enter Character "A". However, if *computerKeyboard* is set up in the QWERTY mode, the computer actor will understand that the user has typed the "Q" key. This simple example shows the importance of representing the state of instruments and of surfaces by the way of interactors so that users can adjust their own representation or act on the configuration of the system to adjust their needs (e.g., switch to the AZERTY mode).

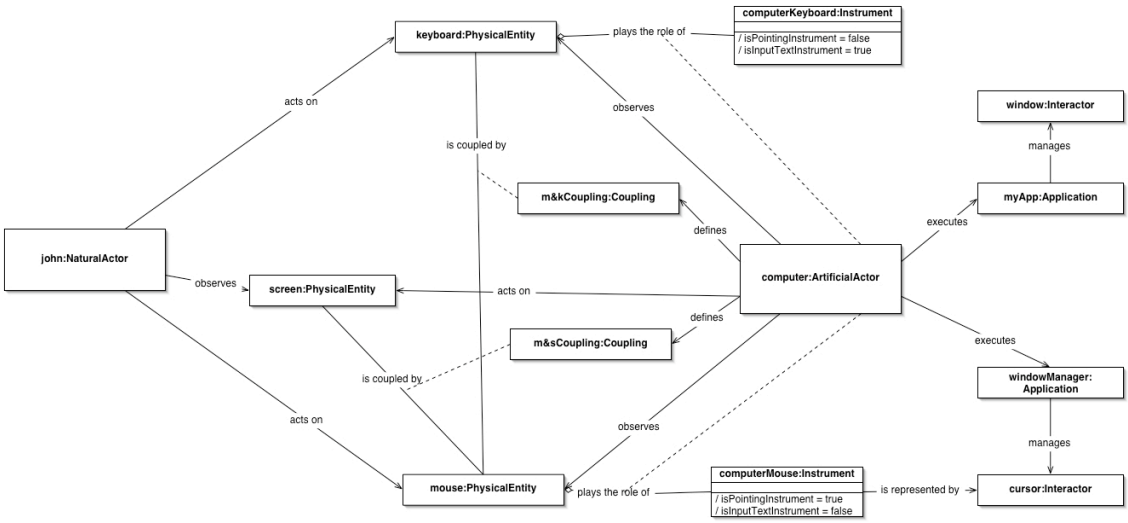


Figure 2.8. In classical GUI, absence of interactors to express the coupling state between the mouse and the keyboard.

One can reason further on this example by analyzing the relationships between the mouse and its representation on the screen, as well as on coupling the mouse and the keyboard used in the GUI paradigm to define the input focus interactor for the keyboard. As Figure 2.8 shows, in classical GUI, there is no interactor to represent the coupling between the mouse and the keyboard. This is acceptable for mono-mouse, mono-keyboard platforms. On the other hand, in multi-surface, multi-keyboard, multi-mouse configurations such as those supported by I-AM, the number of couplings is combinatory. An explicit expression of these couplings is thus mandatory.

3 TECHNICAL PRINCIPLES OF I-AM

I-AM is a technical instantiation of the ontology where I-AM plays the role of an artificial actor. This actor supports the dynamic composition of interaction resources into a unified space. In this space, users can distribute and migrate whole or parts of the user interface as if this user interface were handled by a unique computer. This illusion of a unified space is maintained at no extra cost for the developer. I-AM is a

middleware that can be viewed as an extension of current windowing systems to support the development of multi-surface, multi-instrument interaction in a unified way. It is an enabling technology intended to facilitate the development of user interfaces for smart spaces at the edge of the global infrastructure.

Figure 3.1 shows the developer's view and the user's view of a simple interactive system developed on top of I-AM. From the developer's perspective, the usual GUI programming paradigm is preserved: the programmer creates a window, sets its size and location, fills it with a picture, and asks I-AM to manage the window as an interactor. This interactor is then mapped by I-AM on the physical screens that have been coupled and related in space to play the role of a single large surface. Keyboards and mice are instruments whose representation is mapped on the surface. As a result, users can use any keyboard and mouse to act on any screen.

```
// My program is an IAMApp
IAMApp myiamapp = new IAMApp ();

// Create mywindow, and assign size and location to it
IAMWindow mywindow = new IAMWindow();
mywindow.setCenterLocation (300, 300);
mywindow.setSize (300, 200);

// Insert GLOSS Logo
…
// Ask I-AM to manage mywindow
myiamapp.addInteractor (mywindow);
```

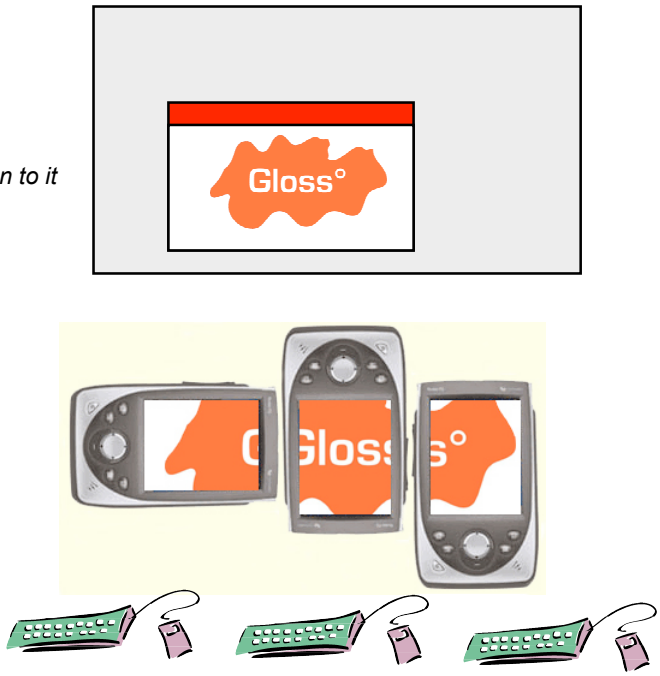


Figure 3.1 The developer's view and the user's view of an interactive system developed on top of I-AM. On the left, the code produced by the developer using the I-AM library. At the bottom right, the user's view where a window, which is distributed across three screens, can be manipulated using any of the mice and keyboards of the cluster.

I-AM advances the state of the art by addressing all of the following problems:

1. Platforms heterogeneity (e.g., clusters of machines running a mix of MacOs X, Windows NT and Windows XP),
2. Interaction resources heterogeneity (e.g., screens with different sizes and resolutions),
3. Platforms and interaction resources discovery based on a fabric of contextors,
4. Multi-surface interaction grounded on the dynamic coupling of hinged display surfaces whose spatial relationships are automatically modeled and maintained,
5. Multi-keyboard, multi-pointer capabilities (so that a user can use the mouse of a PC to manipulate a window displayed on a MacOS screen and drag the window across screens boundaries as if there were a single screen).

Figure 3.2 illustrates the technical principles of I-AM.

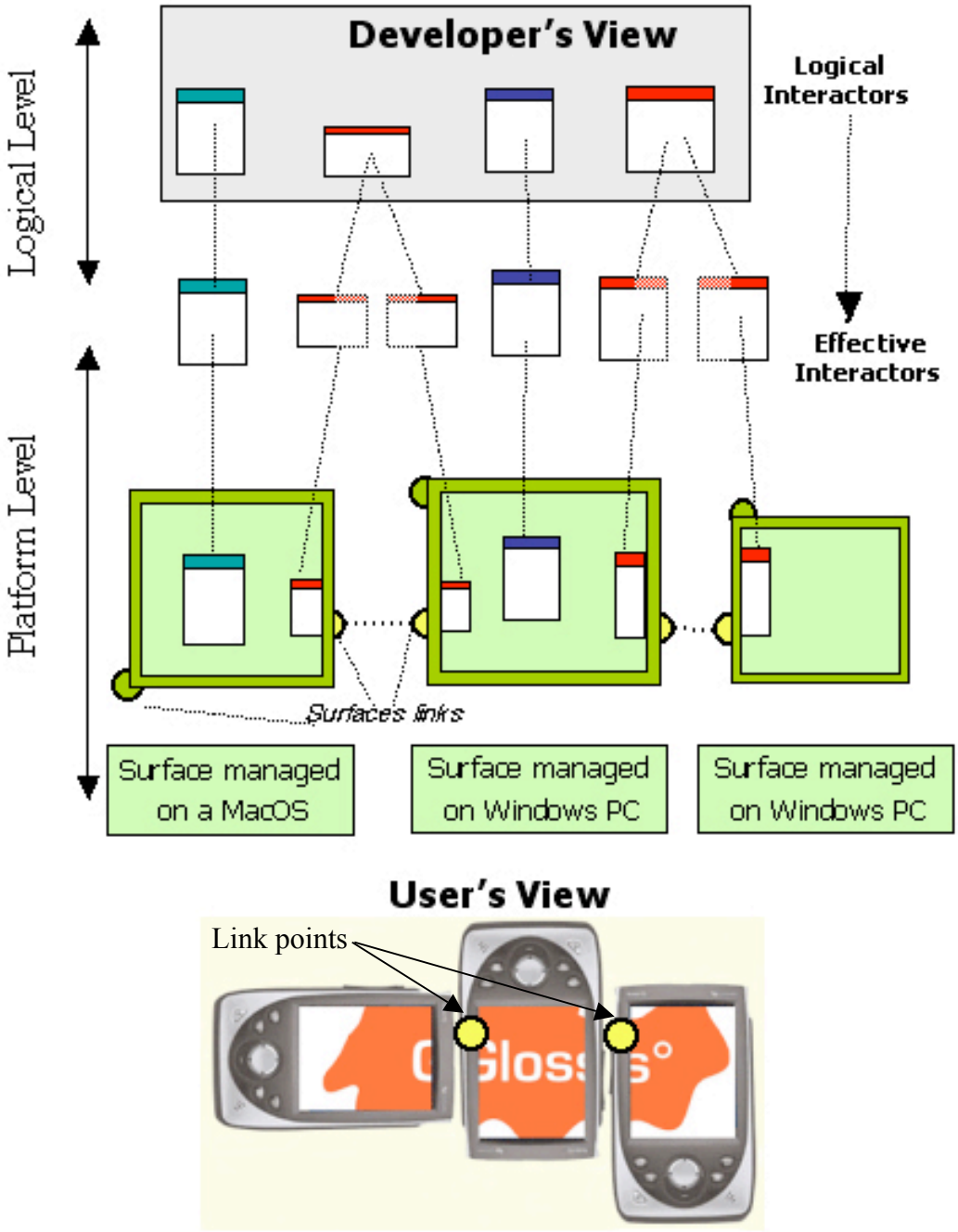


Figure 3.2. The principles of IAM [From D19]. Yellow dots denote link points.

In the example of Figure 3.2, the platform is a cluster composed of three machines that run different operating systems (e.g., MacOS X, Windows XP, Windows NT). Each machine handles a screen and possibly a mouse and a keyboard. Each screen has the role of a surface and each mouse and keyboard has the role of pointing and input text instruments, respectively. Screens have link points. A *link point* is a reference point located at the edge of a screen. It can take the form of a physical sensor (e.g., infrared sensors, accelerometers as in Hinckley's example of synchronous gestures for connecting tablets [Hinckley 03]). It can be a painted dot tracked by a computer vision system, or any conventional identifiable spot on the edge of a screen. Bringing two link points together, from two different screens, results in coupling the two screens to play

the role of a larger surface. Because link points are geometric points, the screens (or surfaces) are automatically related by geometric relationships. In our current implementation, surfaces are rectangular and configured in a plane using, possibly different, orientations in the plane.

The bottom of the figure shows the distribution of the user interface across three screens. Some interactors such as the top left window of the developer's view, are fully rendered within a single screen whereas other interactors, such as the right most window of the developer's view, are split across two screens. In the latter case, the logical interactor of the developer's view is *mapped* into two effective interactors whose rendering is tightly coupled to entertain the illusion of a single surface: as the user moves one of the effective interactors using any pointing instrument of the cluster, the other "twin" effective interactor is moved and resized accordingly as if the twins were one single piece.

By default, the mouse and the keyboard of a machine are automatically coupled by I-AM to provide the "text input focus" function. As a result, I-AM can simultaneously support as many input focuses as there are "pointing instrument-text instrument" couples. However, although any pointer instrument of the cluster can be used to select any text interactor, users must enter text with the text instrument that is coupled to the pointer instrument used to select the interactor. I-AM includes programmable mechanisms to change the coupling between pointing and input text instruments, but these have not been made available yet at the user's level.

The role of I-AM is to continuously track the existence of interaction resources, and to continuously compute the configuration of the screens of the cluster to form a large surface, to dynamically maintain the *coupling* between the instruments and the screens, as well as between the instruments themselves (such as between mice and keyboards), and to maintain the *mapping* between the logical view of interactors as handled by the developer, and the effective interactors as manipulated by the user on the configuration of screens through the set of instruments of the clusters.

In this section, we have briefly presented how I-AM supports the concepts of the multi-surface ontology. A detailed presentation of the implementation can be found in D19. In the next section, we address similar issues from the user's perspective.

4 EXAMPLES OF INTERACTION TECHNIQUES DEVELOPED WITH I-AM

In this section, we present early examples of interaction techniques that allow users to control the interaction space supported by I-AM. As revealed by the ontology, users should be able to control coupling, spatial relationships and mapping.

4.1 COUPLING

Coupling is the act of linking two entities so that they can operate together to provide a set of functions (see Section 2). In the context of I-AM, we present how users can control coupling between screens, between instruments, and between instruments and interactors. These three issues are discussed next.

4.1.1 COUPLING SCREENS

Coupling and decoupling screens are not performed by I-AM but by users: perhaps with some rare exception, users need to control their own interaction space. We have grounded the interaction technique for coupling and decoupling screens on the principles of direct manipulation together with the notion of proximity: to couple two screens, users bring the screens in close contact. To decouple them, they pull them apart. This technique is based on the hypothesis that one screen at least can be hold with hands. For mobile situations, this hypothesis sounds reasonable but needs to be verified in multiple realistic contexts of use.

In order to inform users that a coupling action has been understood by the system and that it is now operational, feedback is provided where users have most likely their current focus of attention, that is, on the screens themselves. In addition, the feedback must express the function that the coupling action provides: that of enlarging the screen real estate. As shown in Figure 4.4, a blue border that outlines the connected screens makes concrete the surface currently available[3].

The screens we are using are not yet equipped with physical sensors. Therefore, we simulate physical contact and/or proximity with an application called the SurfaceConfigurator. The SurfaceConfigurator is used by a human wizard (or by users if they wish so) who mimics users'actions as they bring two screens together. It may run on any computer of the local area network. This computer does not need to be a member of an I-AM cluster. Pictures of Figure 4.1 to 4.6 show a scenario that demonstrates the interaction technique currently available.

Figure 4.1. Initial state of the scenario. Two workstations are running I-AM. Their screens are not coupled. Each one is outlined with its own closed blue border.

[3] This feedback is produced by a component of I-AM: the TopologyRenderer that runs on each of the elementary machines of the cluster (See package IAMPlatform in D19).

Figure 4.1 corresponds to the initial state of the scenario. Two workstations are running I-AM but their screens are currently decoupled: each screen is outlined with a closed blue border.

In parallel, the user interface of the SurfaceConfigurator shows that two screens have been discovered (see Figure 4.2). By default, a screen has the role of a surface. To discover the surfaces of a cluster, the SurfaceConfigurator uses the contextors infrastructure[4]. As shown in Figure 4.2, a surface is represented as a rectangle whose size and borders are proportional to that of the screen it corresponds to. The orientation of the surface in the real world is represented by an arrow. The ID of the surface is displayed on the top left border of the rectangle[5]. The rectangles can be rotated and assembled using the mouse to match the orientation of the screens in the physical world. Clearly, these functions would not be necessary if the screens where equipped with the appropriate sensors.

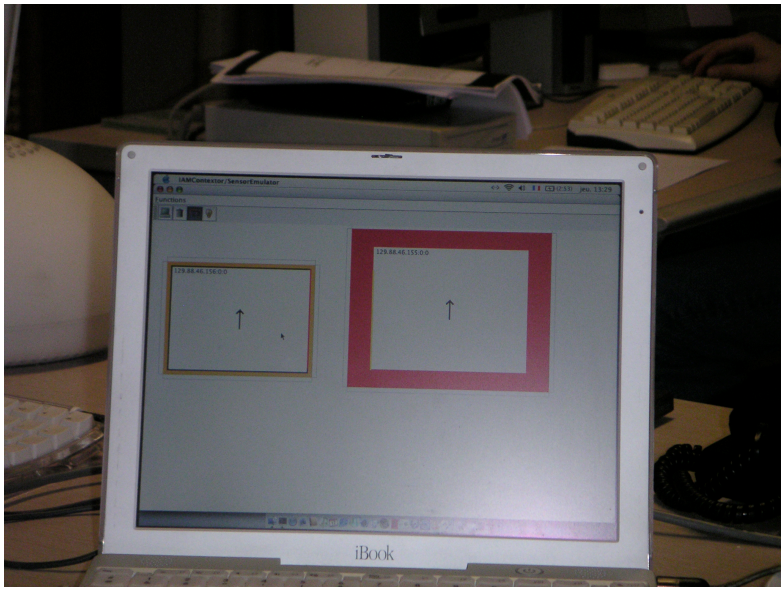


Figure 4.2. The user interface of the SurfaceConfigurator. The interaction space is composed of two decoupled screens.

In Figure 4.3, the user (or the wizard) brings the rectangles and links them side-by-side vertically. When the "computer" icon of the menu bar is selected, the SurfaceConfigurator creates the surfaces links that physically bind the surfaces and, using the contextors infrastructure, publishes the appropriate "Arrival of a New link" events. From there, the two screens are coupled and the appropriate graphical feedback is displayed to concretise the new screen real estate (see Figure 4.4).

[4] As presented in D19, the existence of the surfaces in a cluster as well as their ID and physical characteristics (size, resolution, borders width), are exported to the world by the ContextAdaptor of the PlatformManager that runs on each machine of the cluster.

[5] As presented in D19, the ID of a surface includes the IP address of the machine that handles it, the ID of the graphical port of the video card that handles it, and a unique integer.

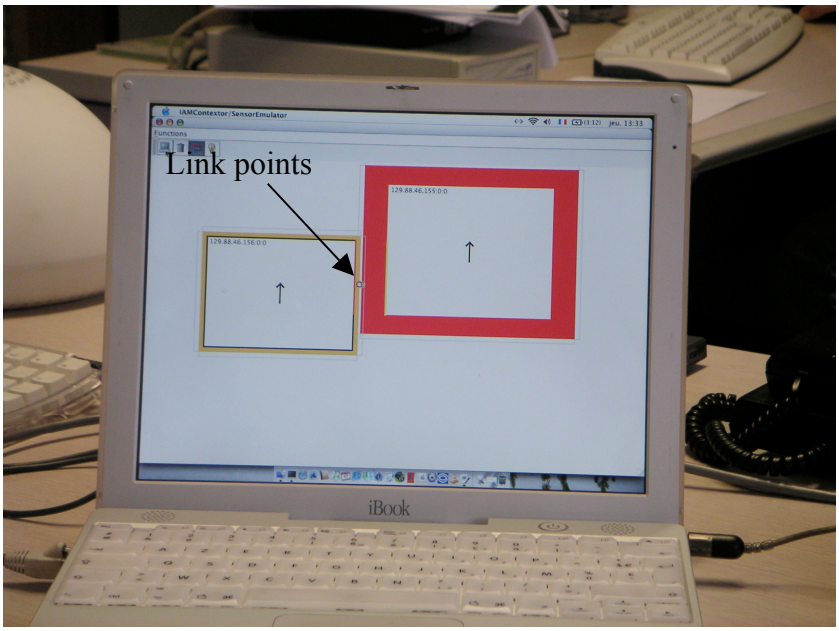


Figure 4.3. Simulation of the action "bringing two screens in contact": the two rectangles are brought together and linked through their simulated sensors whose location denotes link points (Cf Figure 3.2).

In Figure 4.4, the two screens are coupled and a blue border outlines the overall space available for making information observable. At the edges that are common to the two screens, one can observe the "gateway" through which interactors can migrate while preserving their observability. Figure 4.5 shows the "Gloss" window as it overlaps the two screens. The blue-bordered surface behaves as a clipping area: any pixel that sits outside the area is not rendered.

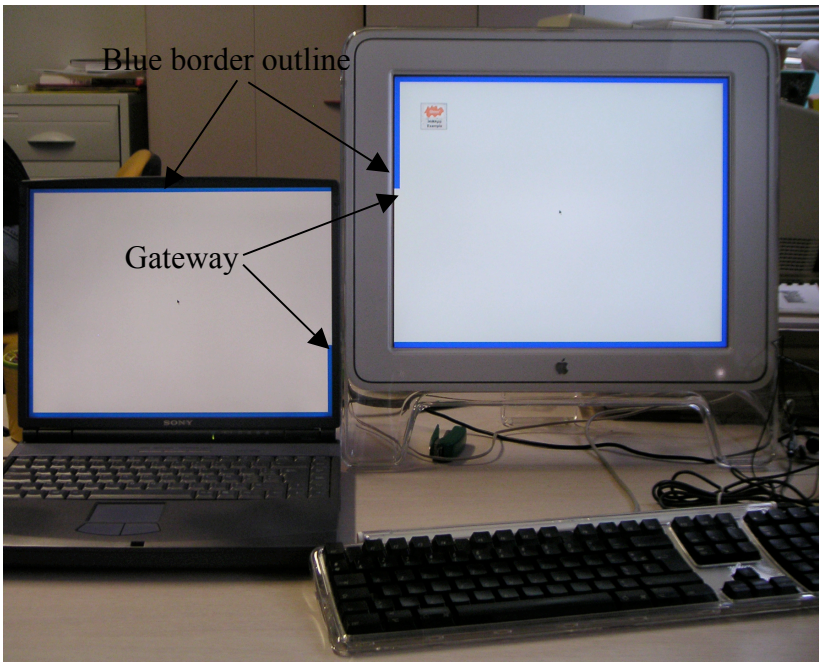


Figure 4.4. The two screens are coupled to provide a larger display area. A blue border outlines the overall space available. At the edges common to the two screens, a gateway shows where interactors can transit while preserving their observability.

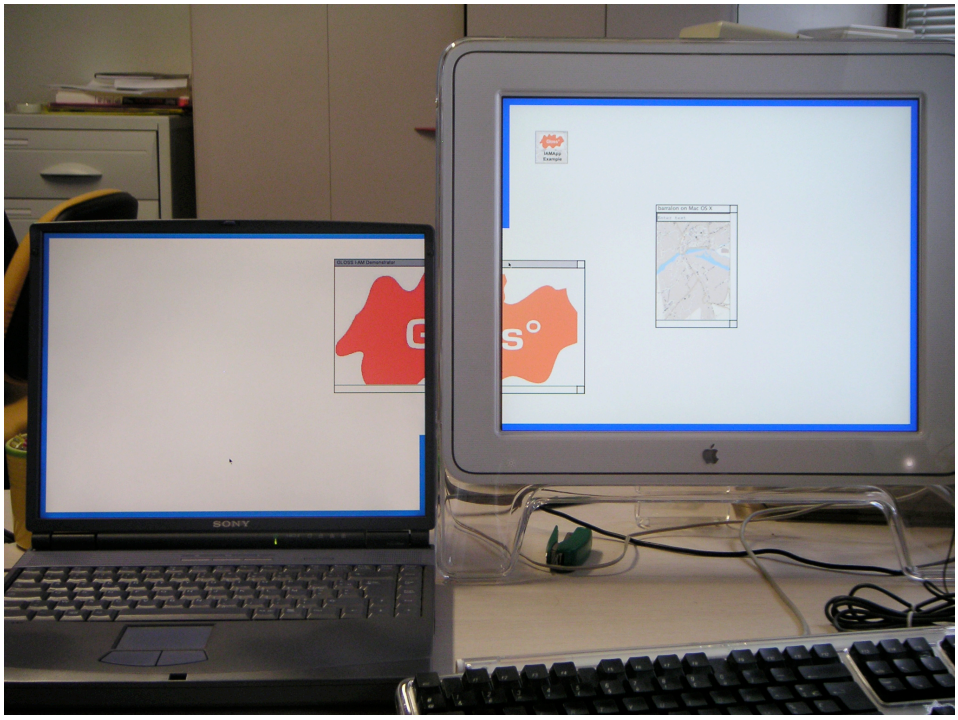


Figure 4.5. A window interactor in transit through the gateway, and dropped between the two screens.

In Figure 4.6, the user has suppressed the surface links between the two surfaces. As a result, the two screens are decoupled and the Gloss window disappears from one of the two screens. The rules that govern the mapping of interactors on surfaces are presented in Section 4.3.

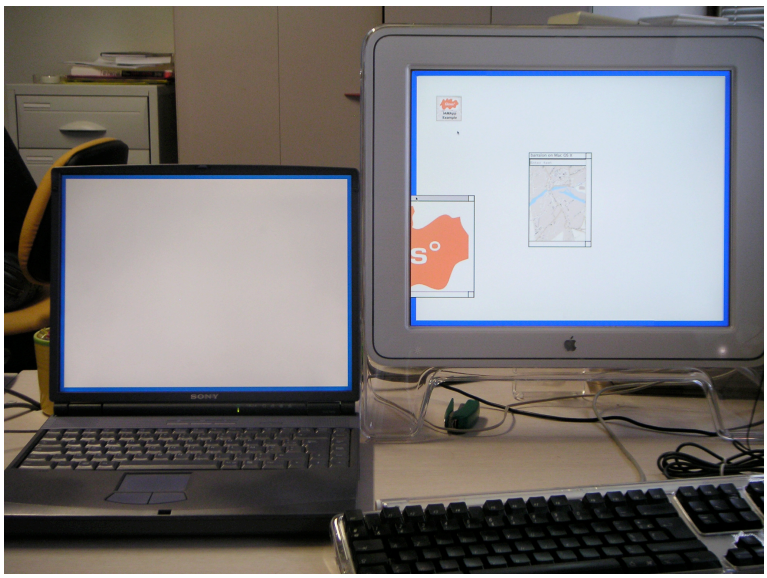


Figure 4.6. The twos screens are decoupled. They now play the role of two independent surfaces. The Gloss window has disappeared from the left screen.

I-AM does not impose any limit on the number of screens that can be coupled. For instance, from the state shown in Figure 4.5, users can couple another screen to the

configuration. As presented in D19, the mosaic of screens play the role of a unique logical surface. Figure 4.7 shows an example of four screens built from a sequence of coupling actions.

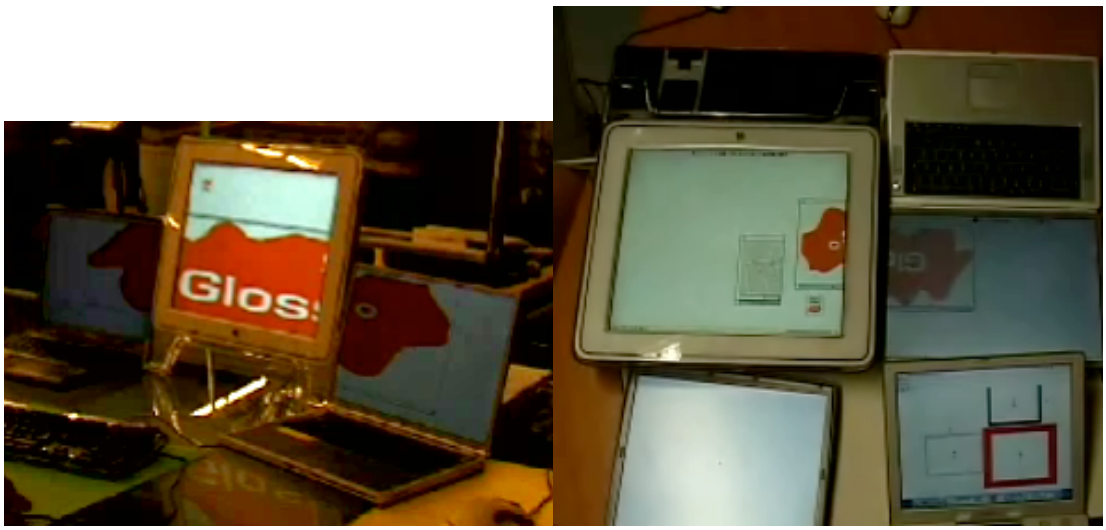


Figure 4.7. Examples of mosaic of screens. Each mosaic plays the role of a unique logical surface.

4.1.2 COUPLING INSTRUMENTS

In the current implementation of I-AM, users cannot define their own couplings between instruments, but we have imagined a number of possibilities.

As presented in Section 2, I-AM automatically defines a coupling between the pointing instrument and the input text instrument of the same machine. However I-AM includes the software mechanism for overriding the default option. Typically, the mechanism makes possible to couple any mouse from any computer with any keyboard from any other computer. This type of coupling may make sense with wireless instruments that can be easily moved in a room. For example, by reusing the interaction technique devised for coupling surfaces, a mouse and a keyboard brought in close contact would couple them to support the input focus function.

As already stated above, I-AM supports as many pointers as there are pointing instruments. In some circumstances, one may want to have a single pointer controllable with multiple pointing instruments. Again, bringing together two pointers controlled by two distinct pointing instruments would bundle them into a single pointer controllable from any of the coupled pointing instruments.

As for coupling screens, feedback must be provided to users such that they can evaluate the couplings they are building or that already exist between instruments. In Section 2, we have shown how the ontology helped us in identifying the absence of interactors to express the coupling between mouse and keyboard in conventional GUI settings. This is fine for static mono-mouse, mono-keyboard platforms. In a flexible dynamic setting, the combinatory nature of couplings between instruments poses a serious problem to be addressed in future research.

4.1.3 COUPLING INSTRUMENTS WITH INTERACTORS

Coupling an instrument with an interactor allows users to change the state of the interactor and thus, to modify the information it represents.

We have not been very creative on this issue: we have reused the typical couplings used in conventional GUI, that is, pointing and selecting and setting the input focus. However, we will see that the multiplicity of instruments leads to interesting situations that need further studies. We illustrate the issue when setting the input focus for text instruments.

Figure 4.8 shows the situation where the user has selected a text field interactor with the touch pad of the PC. Because the touchpad is automatically coupled with the keyboard of the PC (see section 4.1.2), the text field is now the input focus for the PC keyboard. Thus, the user can enter text with the PC keyboard (in the example, "I can type text").

Then, in Figure 4.9, the user selects the text field with the mouse of the Mac. Because the mouse of the Mac is automatically coupled with the keyboard of the Mac, the text field now becomes the input focus for the Mac keyboard as well. The user enters text "here" with the keyboard of the Mac. Since the text field is the input focus for two keyboards, input can be provided simultaneously from the two keyboards.

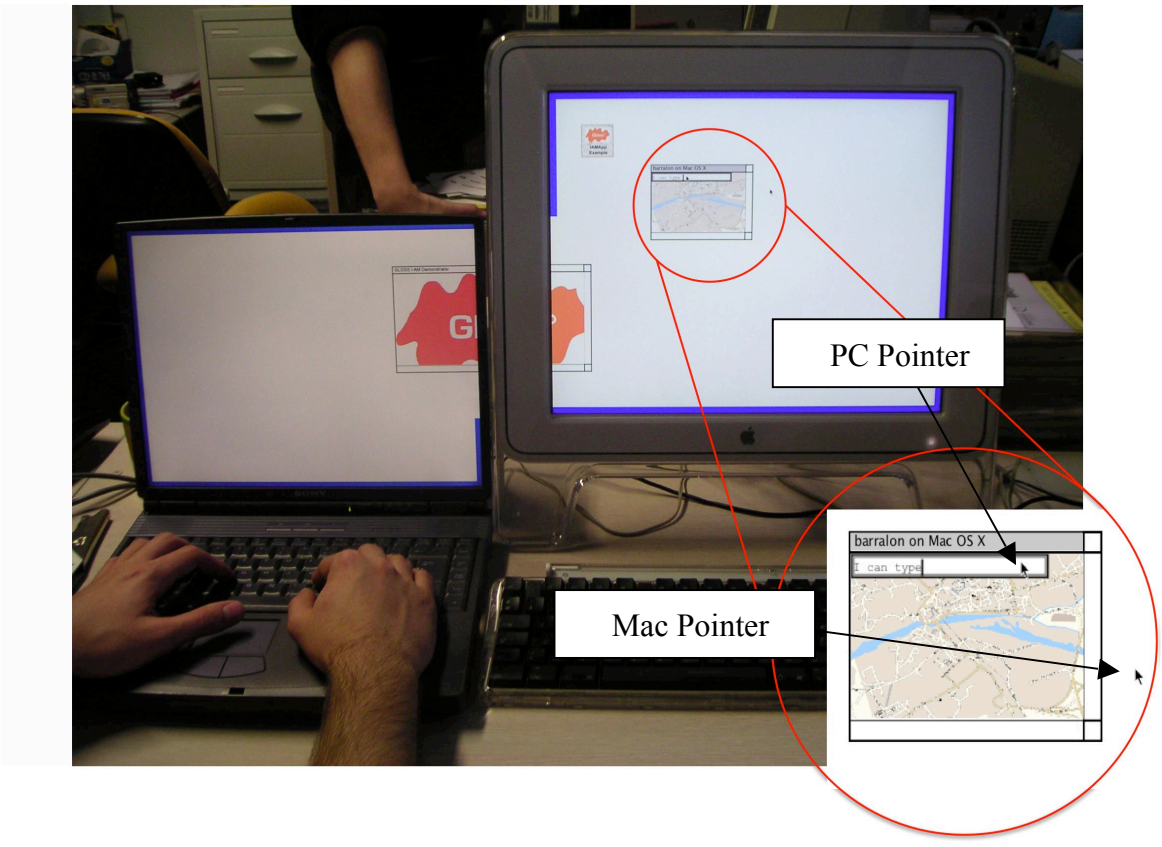


Figure 4.8. The user has selected the text field with the PC mouse and has entered text "I can type" with the PC keyboard.

By generalization, any interactor can be coupled to multiple instruments which, in turn, can be used simultaneously. This facility opens the way to the design of new forms of interaction not possible with conventional GUI toolkits and windowing systems. However, we need to investigate the number and the types of instruments that can be reasonably coupled simultaneously to an interactor.

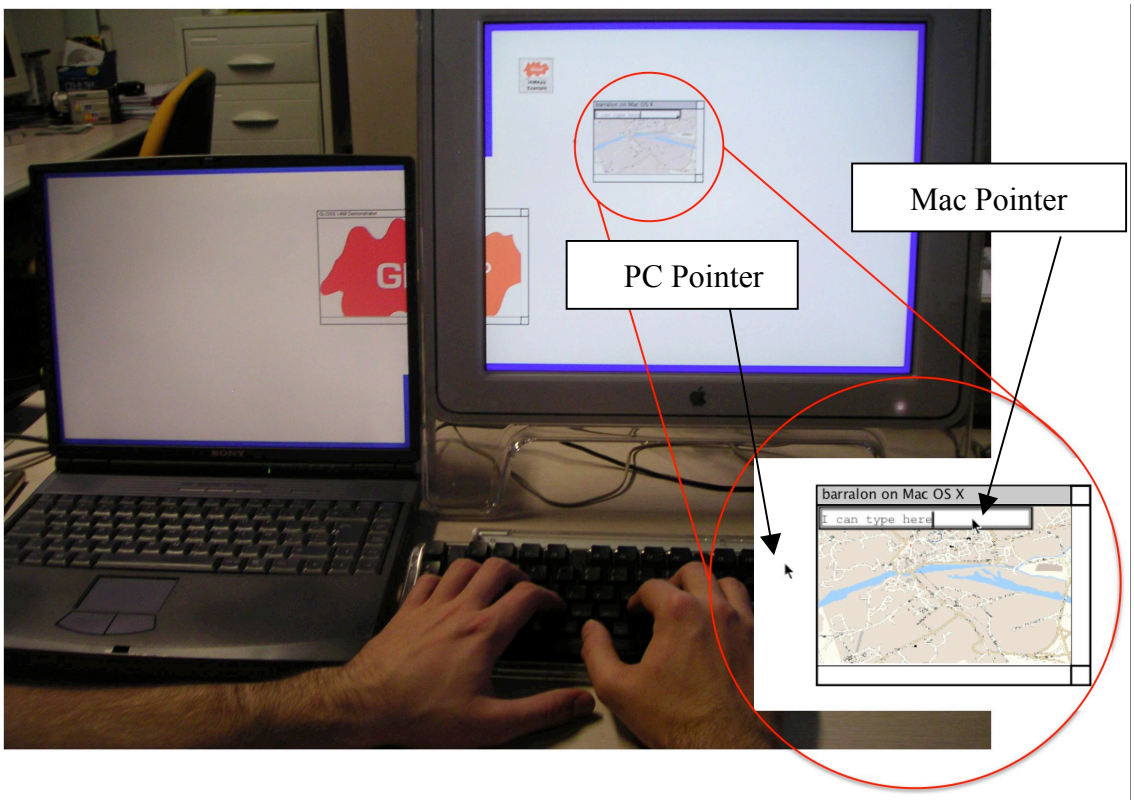


Figure 4.9. The user has now coupled the touch pad to the background of the surface. Then, he has selected the text field with the mouse of the Mac, and has typed "here" with the Mac keyboard. If the user had not changed the coupling set in Figure 4.8, it would have been possible for the user to type from the two keyboards at the same time.

4.2 SPATIAL RELATIONSHIPS

In the current implementation of I-AM, we are far from supporting all of the spatial relationships that the ontology prescribes. I-AM is currently limited to the spatial relationships between screens. As presented above, screens are composed in a plane to form a larger display area. 3D relationships between surfaces must be investigated to give rise to new functions. More generally, spatial relationships between the user and the interaction resources are important issues to address.

Figure 4.10. Top-to-Top composition: two screens are coupled via their top edges.

In this section, we recall the type of screen composition I-AM is able to support (See figures 4.10 to 4.12).

Figure 4.11. Bottom-to-Left composition: The bottom edge of one screen is coupled to the left side of the other.

Figure 4.12. Top-to-Left composition: The top edge of one screen is coupled to the left side of the other.

4.3 MAPPING INTERACTORS ON SURFACES

Our early analysis of the mapping problem is two-fold: the problem of identifying the appropriate mapping metaphor, and the development of appropriate algorithms to maintain visual quality.

4.3.1. MAPPING METAPHOR

As reported by [Hinckley 03], mapping the digital space onto the set of composed surfaces can be performed in many ways. As shown in our examples above or in [Yee 03], one way is to consider each surface as a physical peephole on the digital world. When a new screen is connected, visual access to the digital space is expanded.

Another metaphor is to interpret the arrival of a new screen surface as a way to transform the rendering of the digital content so that it can take full advantage of the new real screen estate. Typically, a city map rendered at a low resolution on a small screen, would be displayed at a high resolution with additional information such as areas of interest, when several screens are docked together. In a multi-user setting, surface contents may be swapped between two users, or joined as in Dynamo [Izadi 03] or the ConnectTable [Tandler 01].

Therefore, the capacity of composing screens opens the way to a large space of design decisions that depend on the application, users activities, and so on. This observation translates into the software design of I-AM by separating the mechanisms from the politics: mechanisms are general so that they can interpret as many application-dependent politics as possible.

By default, I-AM provides the following politics:

- A surface is a peephole through which users access digital content. When a new screen arrives/departs, the peephole is enlarged/shrinked accordingly and shows more/less digital content.
- Each application that runs on the cluster, has its own "reference surface". By default, the reference surface of an application corresponds to the screen that is managed by the machine where the application is launched. If the application is launched from outside the cluster and then is dynamically migrated to the cluster, I-AM provides the application with a default reference surface[6].
- The peephole expands dynamically from the reference surface based on the spatial relationships between the surfaces.
- Every surface that is not coupled (by transitivity) to the reference surface of an application cannot render the interactors of the application.

For example, in the scenario shown in Figures 4.1 to 4.6, the application is launched on the machine that handles the rightmost screen. In 4.5, the peephole is expanded on the left, and the interactors of the application can migrate freely between the two screens (Cf. Figure 4.6). When the screens are decoupled, the left screen, which is not coupled to the reference surface any more, does not render the interactors of the application. As a result, the "Gloss" window is clipped on its left. If the user had moved the window entirely to the left screen, this window would not be accessible once decoupling has occured.

This simple example demonstrates that we, and other researchers, are far away from a new set of appropriate mapping policies along with the appropriate feedback. For the case at hand, we could easily improve the blue border that outlines the available display area, with indices (such as handles) that would provide access to hidden interactors.

4.3.1 MAINTAINING VISUAL QUALITY: THE BEZEL PROBLEM

In the following discussion, we do not promote any algorithm, but we show how bezels can impact the mapping function in order to maintain visual continuity.

From previous work in perceptual psychology, it is reasonable to expect that human performance be influenced by the surfaces topology and the bezels [Campbell 03, Tan 03a]. In [Tan 03b], Tan et al. report a study on the effects of visual separation and physical discontinuities when digital content related to the same task is distributed across multiple displays. Their experiment shows that the physical discontinuities introduced by bezels or differences in depth alone, do not seem to have a significant effect on subjects' performance for text comparison. However, in their setting, windows

[6] The I-AM library allows the programmer to change the reference surface.

content are displayed on a single screen at a time. As discussed below, the mapping algorithms between the digital and the physical spaces produce different results depending on whether the bezels are ignored or taken into account.

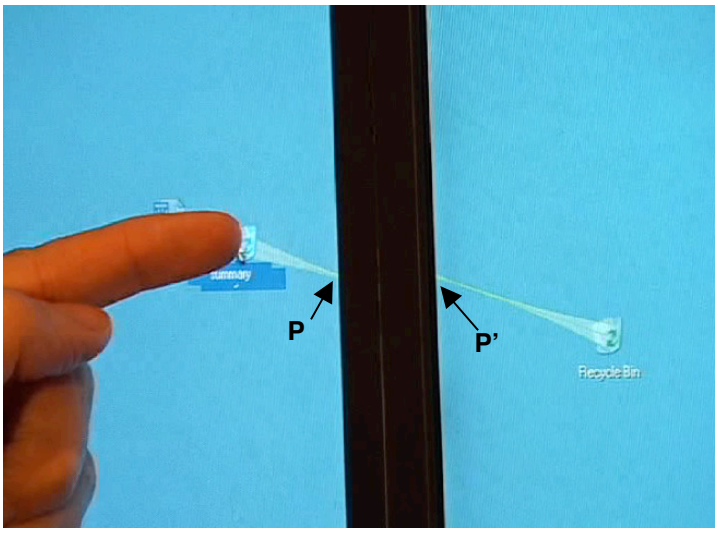


Figure 4.13. A snapshot of the Drap-and-Pop technique developed for the Dynawall for moving icons between remote screens [Baudisch 03]. The mapping technique does not take bezels into account.

The example shown in Figure 4.13, illustrates the effect of bezels. The picture shows the visual effect when the rubber band that joins the base and tip icons crosses the bezels of two contiguous SmartBoards. In this example, the mapping algorithm does not take the bezels into account. Figure 4.14 is a new version of Figure 4.13 that we have modified using Photoshop to maintain visual continuity.

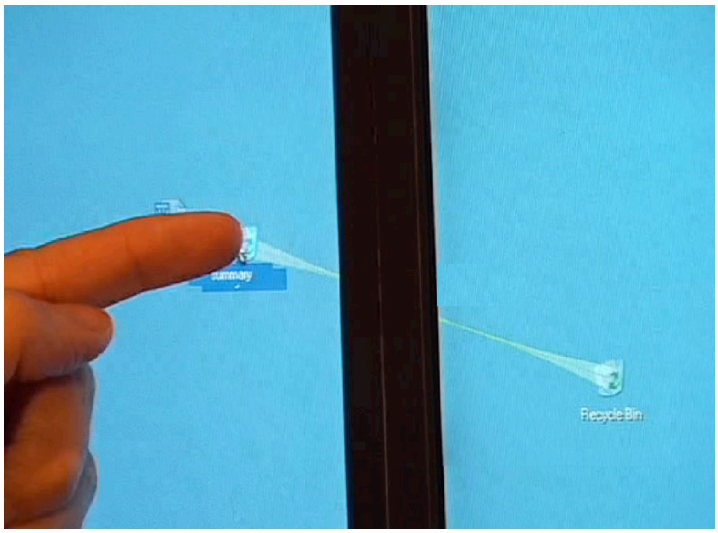


Figure 4.14. The snapshot of Figure 4.13 modified with Photoshop to show how visual continuity may be improved.

The following example illustrates the problem in a more systematic manner. Figure 4.15 shows a physical space composed of three tiled surfaces used to render a digital space that contains the picture of a graph composed of three nodes A, B, C. The top row of the figure shows the final result of the mapping as perceived by users for three different mapping politics. The bottom row shows how the physical surfaces are projected onto the digital content. On the left, bezels are ignored. The result looks like a

broken graph (just like in Figure 4.13). In the middle, bezels are taken into account but are treated as opaque surfaces: the graph looks correct but the pixels that fall under the bezels are lost. On the right, the digital content is processed so that no pixel is lost while preserving the shape of the original image.

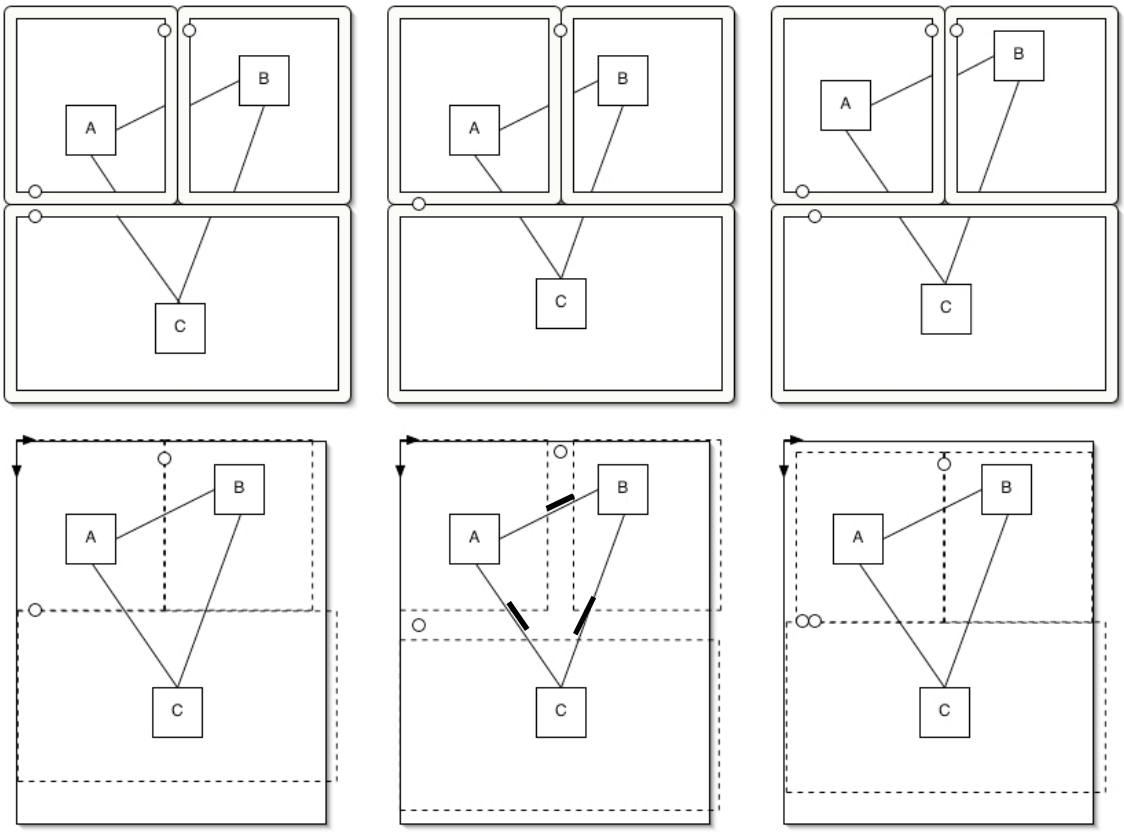


Figure 4.15. Mapping a geometric figure onto three tiled surfaces using different politics. On the left, bezels are ignored. In the middle bezels are modeled as opaque surfaces (at the cost of information loss shown as thick lines). On the right bezels are modeled and the image is modified to improve visual continuity without information loss.

Figure 4.16 illustrates the same problem for rendering mouse cursors. Here, the user is supposed to sit in front of the tiled surfaces. He moves the mouse forward from the bottom surface to the top surfaces. The top left surface is supposed to be the reference, i.e., the surface whose coordinates system is the reference for the topology manager (Cf. D19). On the left, bezels are ignored by the mapping politics. The arrows with dotted lines show the trajectories of the mouse cursor in the digital space. These trajectories are those that the user produces by moving the mouse forward. Thick arrows show the trajectories that the user perceives for each of the politics presented for Figure 4.15. As one can see, it is very hard to maintain the same trajectories in the digital and the physical spaces except for the politics where bezels are considered as opaque surfaces. However, with this politics, the cursor may disappear when it enters the opaque surfaces of the edges.

To summarize, the ontology for multi-surface interaction and its current technical instantiation with I-AM, make explicit a number of important issues for the development of future user interfaces in ubiquitous computing. In the next section, we put them in perspective using Belloti's et al. provocating paper presented at CHI 2002.

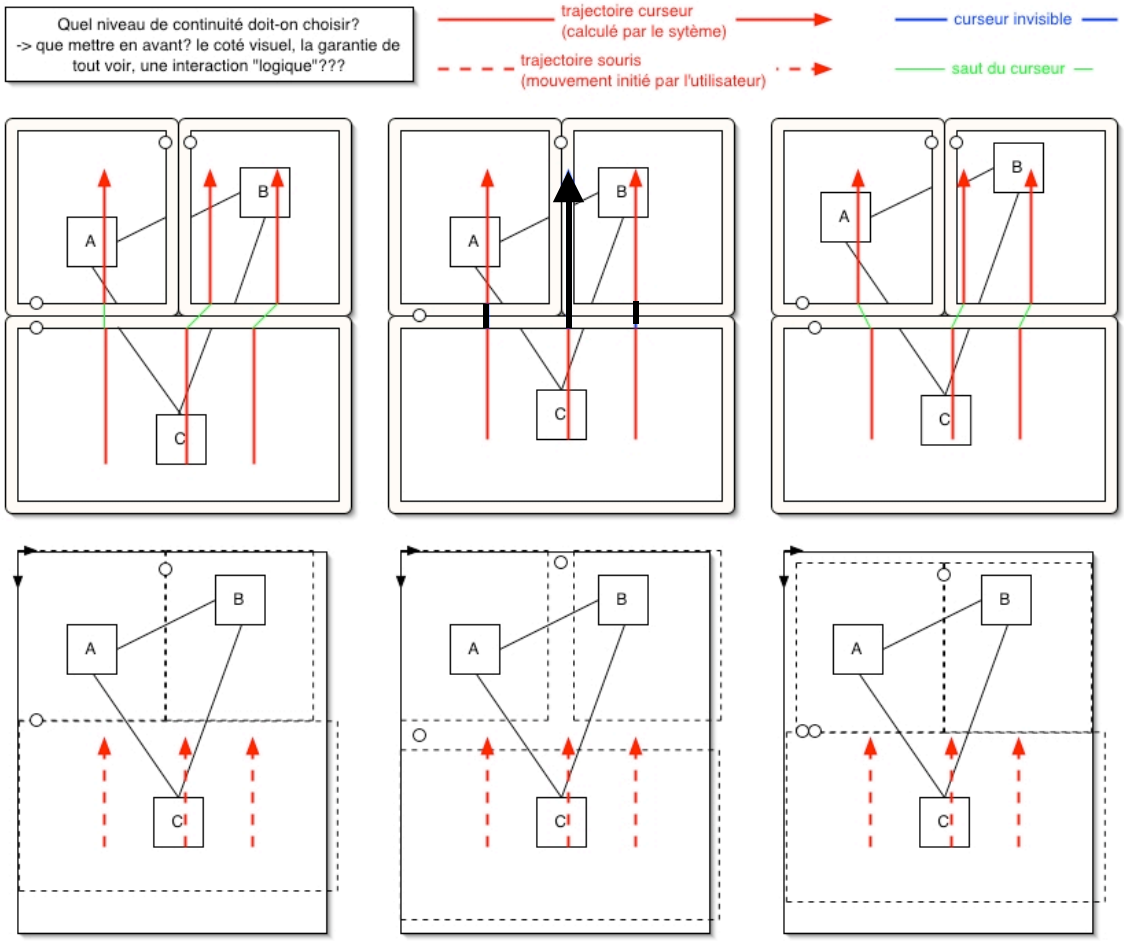


Figure 4.16. Mapping the mouse cursor onto three tiled surfaces using different politics. Thick lines denote cases where the cursor disappears although the user is still moving the mouse. With the politics used on the left and the right, the cursor never disappears. It is however jerky when it crosses the boundaries of surfaces.

5 OPEN ISSUES FOR FUTURE RESEARCH

In their paper, Belloti et al. [Belloti 2002] raise the five following questions for designers and researchers:

1. "When I address a system, how does it know I am addressing it?
2. When I ask a system to do something, how do I know it is attending?
3. When I issue a command (such as save, execute or delete), how does the system know what it relates to?
4. How do I know the system understands my command and is correctly executing my intended action?
5. How do I recover from mistake?"

These questions are inspired from the way social scientists study human-human interaction. In contrast to Norman who stresses the cognitive dimension of interaction (e.g., the user's mental model), Belloti et al. highlight the communicative aspects of interaction. They use the following five issues to cover the same ground as Norman's

seven stage, but with the emphasis on communication rather than on cognition. These are:

1. Address: directing communication to a system,
2. Attention: establishing that the system is attending,
3. Action: defining what is to be done with the system (cf. Norman's gulf of execution),
4. Alignment: monitoring system response (cf. Norman's gulf of evaluation),
5. Accident: avoiding or recovering from errors or misunderstandings.

Figure 5.1 is a remake of the table proposed in [Belloti 2002] that we have extended with a column to present our own problems and solutions. The left most column corresponds to typical basic questions that illustrate the 5 A's communication issues (address, attention, action, alignment, accident). The next column shows examples of well-known answers from the GUI paradigm, followed by two columns that make explicit the challenges and problems raised by the emergence of ubiquitous computing. The right most column corresponds to the solutions that the ontology and I-AM support as well as unsolved problems.

Basic Question	Familiar GUI Answers	Exposed Challenges	Possible Problems	Our problems/solutions
Address: How do I address one (or more) of many possible devices?	Keyboard Mouse (point-and-click) Social control over physical access	How to disambiguate signal-to-noise How to disambiguate intended target system How to not address the system	No response Unwanted response	Explicit coupling + reuse of familiar GUI answers (point-and-click, social control access)
Attention: How do I know the system is ready and attending to my actions?	Graphical feedback (e.g., flashing cursor, cursor moves when mouse moved) Assume user is looking at monitor	How to embody appropriate feedback, so that the user can be aware of the system's attention How to direct feedback to zone of user attention	Wasted input effort while system not attending Unintended action Privacy or security problems	Graphical feedback. Same as GUI answers. Pb in a multi-user setting: How do I know the system is ready and attending MY own actions? Is the system able to observe multiple users acting on the same screen at the same time? If so, how to direct feedback to a particular user without disturbing the other users?
Action: How do I effect a meaningful action, control its	Click on objects(s) or drag cursor over area around object(s). Select	How to identify and select a possible object foraction.	Limited operations available	Interaction techniques for coupling: what is (are) the appropriate metaphor (s) for

extent and possibly specify a target or targets for my action?	objects from menu (e.g., recent files). Select actions from menu, accelerator keys, etc. Manipulate graphical controls (e.g., sliders).	foraction. How to identify and select an action, and bind it to the object(s) How to avoid unwanted selection. How to handle complex operations (e.g., multiple objects, actions, and more abstract functions that are difficult to represent graphically, such as save).	Failure to execute action Unintended action (wrong response)	coupling surfaces, for coupling instruments, and coupling instruments and interactors (digital content)? Cf. Hinckley's sunchronous gestures to couple tablets [Hincley 03] or Rekimoto's SyncTap What is the impact of the physical characteristics of interaction resources on the nature of the interaction techniques? For example, how GUI techniques apply when interacting in the large and in the small? Cf. Baudisch Drag-and-Pop and Drag-and-Pick [Baudisch 03], the GroupBar as a new task bar for large displays [Smith 03], the high density cursor to support fast moving mouse on large displays [Baudisch 03c], or Halo to support spatial cognition on small screens [Baudisch 03b] What is the impact of 3D spatial relationships on the nature of interaction techniques? How to assign roles (e.g., surface and instruments) to interaction resources? Interaction techniques for navigating within the information space: what is the appropriate mapping metaphor (s) between the logical space and the interaction resources? How to distribute the user interface components across the diversity of interaction resources (e.g., migration of information between

				surfaces). How to specify that some resources are public while others are semi-public or private?
Alignment: How do I know the system is doing (has done) the right thing?	GUI presents distinctive graphical elements establishing a context with predictable consequences of action Graphical feedback (e.g., characters appear, rubberbanding) Auditory feedback Detectable new state (e.g., icon in new position)	How to make system state perceivable and persistent or query-able How to direct timely and appropriate feedback How to provide distinctive feedback on results and state (what is the response) Inability to differentiate more than limited action space Failure to execute action Unintended action Difficulty evaluating new state	Inability to detect mistakes Unrecoverable state	What is the nature of feedback to express existing couplings? In particular, how do I know that a particular interactor is coupled to this set of instruments? What is the nature of feedback to express that two entities are compatible and therefore can be coupled, or are being coupled, or are on the point to be coupled (e.g., there are getting close to each other)? What is the nature of the feedback to express the availability of the new set of functions that results from coupling? What is the nature of feedback to express that the user interface is currently migrating between interaction resources and/or is being plastified to adjust to the new interaction resources? In a dynamic multi-computer setting, where do my data sit? For example, when dragging and dropping a file icon from one surface to another, what is the location of the file? In other words, is the drag and drop metaphor used in traditional desktops still valid for clusters. In a multi-user setting, how do I know that this particular feedback corresponds to my own actions, not to my peer's actions

Accident: How do I avoid mistakes ?	Control/guide in direct manipulation Stop/cancel Undo Delete	How to control or cancel system action in progress How to disambiguate what to undo in time How to intervene when user makes obvious error	Unintended action Undesirable result Inability to recover state	Forward recovery: Decoupling and de-migrating functions

6 CONCLUSION

We have defined a new concept for interacting in global smart spaces: that of multi-surface interaction. This concept relies on a known model, the Model of the Human Processor, where the "cognition-actuator-sensor" structure of a human actor has been extended with a symmetric counterpart (the artificial world), and where the artificial and human actors mediate through interaction resources: surfaces and instruments. These resources are composed in an opportunistic way by users to form a dynamic interaction space. We have developed I-AM, a middleware infrastructure that supports this vision.

We have developed an initial set of interaction techniques that allow users to configure their interaction space. These techniques have not been evaluated through formal user studies. Instead, they have served as a prospective apparatus to explore the problem space and test the soundness of the technical solution. In particular, we have re-used known solutions from the GUI paradigm. It is our intention to be less conservative and push our research forward now that we have a running infrastructure available.

7 REFERENCES

[Baudisch 03] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., Zierlinger, A. Drag-and-Pop and Drap-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems. In Proc Human Computer Interaction –INTERACT'03, Rauterberg et al. Eds., IOS Press Publ., IFIP, 2003, pp. 57-64.

[Baudisch 03b] Baudisch, P., Rosenholtz, R. "Halo; A technique for Visualizing off-screen locations". In Proc. CHI'03, pp. 481-488.

[Baudisch 03c] Baudisch, P., Cutrell, E., Robertson, G. "High-Density Cursor: a visualization technique that helps users keep track fast moving mouse cursors". In Proc. Interact'03,2003.

[Bérard 03] Bérard, F. The Magic Table: Computer-Vision Based Augmentation of a Whiteboard for Creative Meetings, in IEEE workshop on Projector-Camera Systems (PROCAM), CD-ROM proceedings of the International Conference on Computer Vision (ICCV), Nice, France, Oct. 2003.

[Campbell 03] Campbell, C., Maglio, P. Segmentation of Display Space Interferes with Multitasking. In Proc Human Computer Interaction –INTERACT'03, Rauterberg et al. Eds., IOS Press Publ., IFIP, 2003, pp.575-582.

[Norman 86] Norman, D., Draper, S. *User Centered System Design*, Lauwrence Erlbaum Ass., Pub., 1986.

[Norman 99] Norman, D. "Affordance, Conventions and Design", *Interactions*, ACM Publ., pp. 38-42, May-June 1999.

[Tan 03a] Tan, D. Gergle, D., Scupelli, P., Pausch, R. With Similar Visual Angles, Larger Displays Improve Spatial Performance. In Proc. Computer Human Interaction –CHI 2003, ACM Publ., Fort Lauderdale, April 5-10, 2003, pp. 217-224.

[Tan 03b] Tan, D., Czerwinski, M. Effects of Visual Separation and Physical Discontinuities when Distributing Information across Multiple Displays. In Proc Human Computer Interaction –INTERACT'03, Rauterberg et al. Eds., IOS Press Publ., IFIP, 2003, pp. 252-255.

[Hinckley 03] Hinckley, K. Synchronous gestures for multiple persons and computers. Proc. UIST 2003, ACM, 2003.

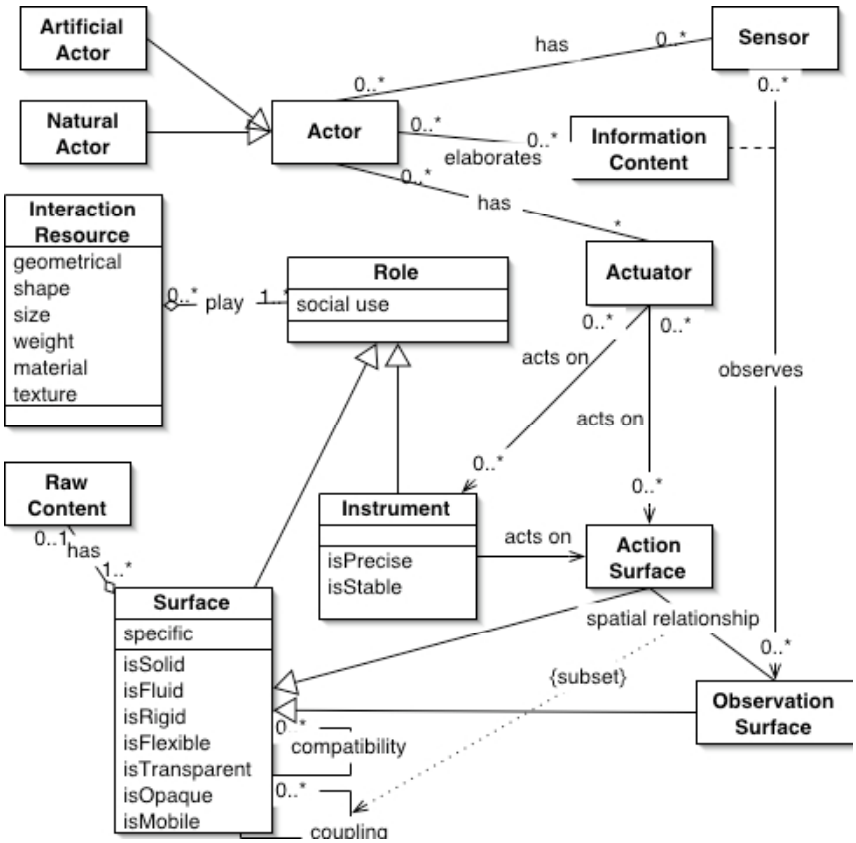
[Izadi 03] Izadi, S., Brignull, H., Rodden, T., Rogers, Y., Underwood, M. Dynamo: A public interactive surface supporting the cooperative sharing and exchange of media. In Proc. UIST 2003, ACM Pub., 2003.

[Smith 03] Smith, G., Baudisch, P., Robertson, G., Czerwinski, M., Meyers, B., Robbins, D., Andrews, D. GroupBar: The TaskBar Evolved, in Proc. Of the ACM CHI conference, CHI 2003, ACM Publ.

[Tandler 01] Tandler, P., Prante, T., Müller-Tomfelde, C., Streitz, N., Steinmetz, R. ConnecTables: Dynamic Coupling of Displays for the Flexible Creation of Shared Workspaces. In Proc. UIST 2001, ACM publ., 2001, pp. 11-20.

[Yee 03] Yee, K. Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers. In Proc. Computer Human Interaction –CHI 2003, ACM Publ., Fort Lauderdale, April 5-10, 2003, pp. 1-8.

8 ANNEX1. MULTI-SURFACE ONTOLOGY: PREVIOUS UML DIAGRAM



This diagram makes explicit coupling and spatial relationships between surfaces only. Mapping is missing. In our new diagram, the relationships apply between any physical entities.