Ad-hoc Co-located Collaborative Work with Mobile Devices

Ad-hoc Co-located Collaborative Work with Mobile Devices

Kris Luyten Kristof Verpoorten Karin Coninx

Hasselt University
Expertise Centre for Digital Media
and transnationale Universiteit Limburg
Wetenschapspark 2
3590 Diepenbeek (Belgium)

{kris.luyten,kristof.verpoorten,karin.coninx}@uhasselt.be

ABSTRACT

This paper presents how ad-hoc co-located collaborations can be supported with an arbitrary number of users that only have acces to small-size mobile displays. Our approach is based on tracking of these personal displays that share the same information space. All displays involved in the collaboration act as autonomous windows on a set of data items (the information space) positioned on a shared virtual canvas. Data items are identified by their three-dimensional location in physical space and can be manipulated through the displays that serve as windows on the shared canvas. Each display is tracked in physical space and is aware of its own location. Since different mobile displays can access and manipulate the same information space, a distributed locking mechanism makes sure the data stays consistent during simultaneous access of data in this information space.

1. INTRODUCTION

The increasing mobility and design of computing devices offers new ways to collaborate with others. Mobile collaboration is still not supported sufficiently to allow people to have "on the spot" meetings where they cooperatively execute tasks and manipulate shared information spaces through their mobile devices. First, the screen space available on current (and probably future) mobile systems is too limited to display large or complex information spaces for everyday usage. Even if the mobile devices are able to provide a usable visualization of a large information space, current user interaction techniques to navigate through this space are insufficient (e.g. buttons on mobile phone or the stylus on a touch screen). In particular current interaction techniques for navigation and manipulation of a multi-dimensional information space on mobile devices are far from optimal.

The following scenario will give an overview of what type of problems the system presented in this paper helps to overcome. Suppose several persons working on an architectural project gather at the construction site to meet and discuss the rough sketches for the building. The purpose of this meeting is to establish a consensus about the final style of the building for this construction site and collaboratively edit and annotate the current draft sketches. The project leader has already created a graphical presentation of the most important parts of the building and has his sketches stored on his laptop. He wants to show his ideas to all other team members, and he also wants them to make annotations and suggestions to improve the design of the sketches. Unfortunately only one person can control the laptop to change parts of the drawing or to write down his/her suggestions next to it. Because of this, either one person has to make all the changes other people tell him/her, or everyone has to wait their turn to work on the laptop. Furthermore, since the meeting described here is a "mobile" meeting in the sense that it is not held at a workplace but by people in an arbitrary place, the limited resources that are available to mobile users for collaboratively manipulating a digital representation of any kind should be taken into account. The only constraint for the system we present in this paper is the requirement of co-located users.

In this paper we show that distributed mobile peephole displays can be used to overcome the limitations imposed by ad-hoc, "on-the-spot" or on-site meetings. Such a set of displays allows every participant to share, view and manipulate data during the meeting. For this purpose, we developed the portable personal displays (PPDs) library *pizu* (pronounced "paay tsu") that enables developers to build applications that go beyond the stationary desktop-based systems for collaboration. When using PPDs in a meeting, every participating member uses his or her own mobile device, given it has graphical capabilities that are sufficiently powerful. PDAs, tablet PCs or modern mobile phones are clients that can use *pizu*-based applications. Every one of these devices has its own view on the shared information space. For example, when the project leader from our previous scenario adds his drawing to the shared information space, all other people in the meeting can study and manipulate the shared drawing at once.

The remainder of this paper is structured as follows: Section 2 describes some related work. Section 3 covers the general hardware requirements of our system. Next, section 4 explains the software architecture. An important issue when using a shared information space is data consistency, our solution to this problem is discussed in section 5. We also developed two example applications with *pizu*, which

are introduced in section 6. And finally section 7 contains a discussion and section 8 provides the conclusion.

2. RELATED WORK

Several solutions to overcome the limited screenspace can be found in literature. The Boom Chameleon [23], introduced by Fitzmaurice and based on the Chameleon [10], is a handheld display developed to browse large or complex information spaces. This system uses a screen attached to a tracking device commonly used for BOOM setups. The objects contained in the information space have a location in the physical environment. By moving the screen around in physical space, a user can observe and interact with these objects through the display. This has proved to be a very natural way for navigating a large and complex information space that does not scale well to the physical screen size, since the virtual information itself gets a physical dimension by the use of screen tracking. The example provided by Fitzmaurice et. al shows how a 3D model of a car can be examined into detail by moving the display around as if the car were parked right in front of the user. Unfortunately this setup is not portable and the BOOM limits the freedom of movement. The system is also intended for individual usage. It is usable for professional applications inside a workplace but hardly usable when a mobile group of users needs to use a large and complex information space.

Ka-Ping Yee's peephole display implementation combines two-handed interaction with spatially aware portable displays [26]. It is an extension of the approach Fitzmaurice proposed, but implemented on mobile devices. The physical position of the display is tracked and used to navigate the information space. The usage of a mobile device in this setup allows the user to navigate the information space using only one hand, which leaves the other hand available for manipulation of the information space through the display. Both 2D and 3D navigation are possible, 3D navigation includes zooming by moving the display vertically. The applications presented in [26] where developed with a single user in mind. The work presented by Ka-Ping Yee has heavily influenced the work presented in this paper. There is no support for collaboration with other users sharing the same information space however. Since this is a natural extension that is useful in various situations we will explore the collaborative aspect of such a system in this paper.

Following the generalized peephole metaphor introduced by Butz and Krüger in [7], the displays in our system will act as multiple peepholes that provide both input and output capabilities on a continuous virtual surface. Different from the generalization presented in this work where an instrumented environment equipped with steerable projectors ("output peepholes") and tracking cameras ("input peepholes") are used, our work does not use such a heterogeneous ensemble of input/output devices. Although our system provides opportunities to act as collaborative tools in augmented reality environments, we explore its usage for everyday applications outside an instrumented room. Similar with what is presented in [7] an infrastructure to track the peephole displays is required. Section 3 provides more detail of our setup.

3. HARDWARE SETUP

The system described in this paper relies on a set of hardware components that need to be available. The following three components make up all the required hardware to set up a meeting:

- wireless 3-dimensional location tracking,
- small displays with a high update rate,
- wireless communication without the need for a wireless access point

The first component is the most challenging one. The handheld displays need to know their own position in physical space, therefor 3D tracking is required for our purposes. During the last couple of years, many experimental systems have been developed that support 3D tracking [4, 19, 21, 22, 24], but nearly all of them are static in the sense that they require a fixed configuration that is far from mobile. We used the V-scope system which is developed by Lipmann Electronic engineering. Figure 1 shows the system: it is an acoustic "time-of-flight" system. Small buttons are provided that are tracked by the V-scope and that can be attached to physical objects. Figure 5 shows two PPDs that each have a button attached on top of them. The V-scope uses three towers that send infrared signals to the buttons, and the buttons reply by sending an ultrasonic signal back to the towers. Because the speed of sound is known beforehand and the time in between sending the infrared signal and the reception of the ultrasonic signal are known, each tower can calculate how far away the button is located from itself. Next, because the distance between the three towers is known, triangulation can be used to calculate the position of the buttons (figure 1).

The biggest advantage of the V-scope is that it is a wireless system, with a reasonable large working range (up to five meters). Futhermore, it can track up to four light-weight buttons with great accuracy. The main disadvantage of the V-scope is that there needs to be a line-of-sight between the tracked button and the three towers. Another disadvantage is that only the position of the button is tracked. If the orientation in 3D space is also required, we need to attach three buttons to one object and calculate the orientation of the object from the positions of the three buttons in space. An alternative approach is to use a set of sensors such as the ones used for the Microsoft XWand [25]. Although the V-Scope is clearly not mobile enough, it is the solution that currently comes closest to a mobile 3D tracking system.

The remainder of the hardware that is used for the system is made up of fairly standardized components. PDAs with Bluetooth support and that can run the Microsoft .Net Compact framework are sufficient to offer the user wireless communication without the need for a wireless access point [6]. Most PDAs are nowadays equipped with a small display with a fairly good response time. Since Bluetooth enables communication of at least 10 metres (and up to 100 metres) this is sufficient for our purposes. In an ad-hoc mobile meeting, supported by our system, users are in the same physical space and there is rarely more than 5 metres in between them while they are working together on a shared information space.

4. SOFTWARE ARCHITECTURE

4.1 Shared and Distributed Objects

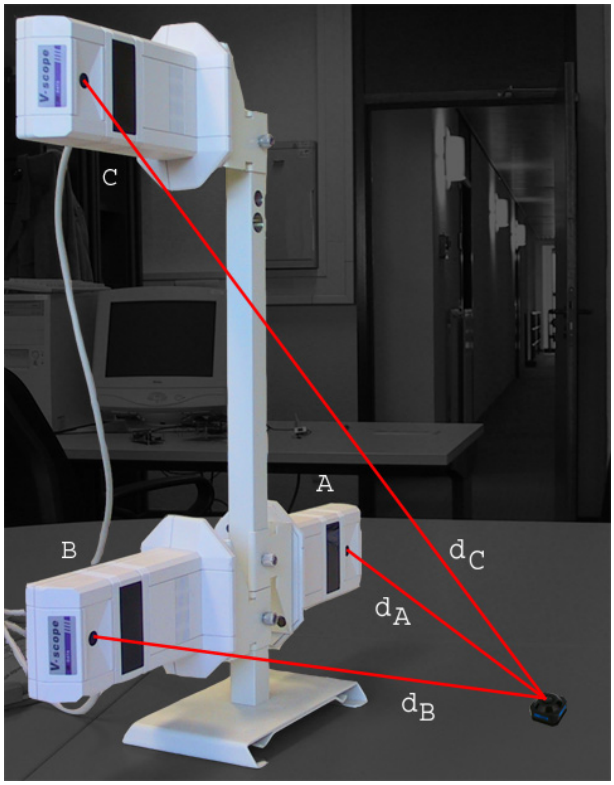

Figure 1: The distances between the button and each tower (A, B and C) are measured (d_A, d_B and d_C). With these three distances and triangulation the position of the button can be calculated.

Different people have to be able to work simultaneously with the same data, so all data has to be shared between the PPDs that participate in the collaborative work session. Because there is no personal computer that functions as a central server, the shared information space has to be truly distributed over the participating PPDs. This essentially creates the possibility for ad-hoc collaboration between the displays, which also brings along some problems regarding data consistency and coordination of updates to maintain the consistency.

The library we created to support ad-hoc collaboration with mobile devices works on top of the Microsoft .Net Compact Framework[1]. It makes extensive use of the .Net version of the Piccolo toolkit[2] for zoomable user interfaces [2, 3, 13, 14, 18] provided by the Human-Computer Interaction Group of the University of Maryland. Because of this, all applications developed for PPDs with our library can create a rich graphical user interface with support for semantic zooming. Figure 2 shows three pictures of a PPD's view on a shared information space that contains several images. The main difference between the subfigures 2.1, 2.2 and 2.3 is the zoom level which is obtained by moving the PPD closer or farther away from the user. In figure 2.1, the PPD is held closer to the user than in figure 2.2. Figure 2.3 shows the view when the PPD is held even further away from the

[1] `http://msdn.microsoft.com/netframework/`
[2] `http://www.cs.umd.edu/hcil/piccolo/index.shtml`

Figure 2: This figure shows three pictures of a PPD's view on a shared information space. The information space contains several images. The main difference in the PPD's physical positions shown in the figure are their heights. In subfigure 1, the PPD is held higher than in subfigure 2. Subfigure 3 shows the view when the PPD is held even lower. This change in height has the effect of zooming in onto the shared information space.

user. This change in distance with respect to the user has the effect of a zooming operation on the visualization of the shared information space.

The library contains a class `SharedObject` that represents the base functionality of objects contained in the shared information space. It is an abstract base class for all objects that can be shared between different devices: it is essentially an extension of a Piccolo Node with support for shared usage in collaborative distributed systems. A Piccolo Node is the central class of the Piccolo toolkit; all objects that need to be visualized on screen are instances of this class. In section 5 we will discuss the locking mechanism provided by the library, which is essential for the correct execution of a highly interactive distributed system.

The *pizu* library also contains the `SharedImage` (any image file) and `SharedPath` (geometrical objects e.g. rectangles, lines, ...) classes that are derived from `SharedObject`. These two classes have a graphical presentation in the user interface of the application that is developed with *pizu*. They are very useful for any kind of basic graphical interface. Anyone can add new kinds of distributed objects by deriving their class from the `SharedObject` class. It is possible to compose different objects of type `SharedObject` in a hierarchy in a Composite pattern [11]. Every `SharedObject` can have children of type `SharedObject`. A child will move the same distance when its parent is moved, and will be removed when the parent is removed. Locking a parent object does not affect the child objects unless explicitly stated so; child objects can still be moved or erased independently of the state of the parent object, unless they are also locked.

All objects have a fixed position in space (unless they are being moved by the user), and usable space is not limited to the display of the user, but is constrained by the range of the tracking technology. To see objects that are out of reach of the current display position, the user can move their display to another place in the physical space to see other objects that are located there. Figure 3 shows this concept: the information space contains a map in this example. Each display shows a portion of the map and can be moved in

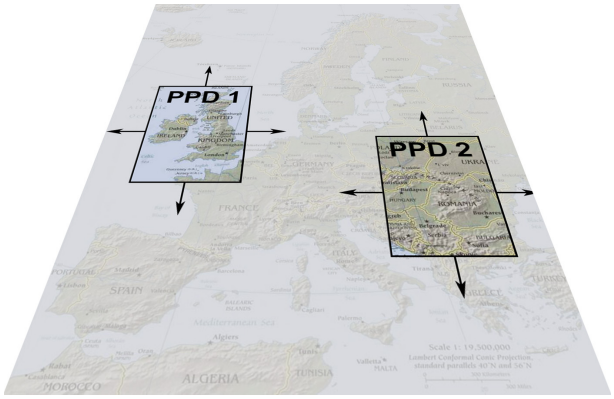

Figure 3: Two PPDs showing each a portion of a shared information space.

physical space to show another portion of the map. This is very similar with a toolglass [5]. Notice the display of the mobile device is now acting as an output peephole on the shared space. When a user adds a shared object to the shared information space, all other users can see this object by moving their display to that position.

It is also possible to move a shared object together with a display. E.g. when a user holds the object with his/her stylus, and moves the mobile device (and thus the display), the object's position will be fixed with respect to the display's position (figure 4). In this case the display acts as an input peephole. Since our library is an extension of the Piccolo library, the same functionality is supported but extended with support for multiple input/output peepholes.

4.2 Networking

Mobile devices that want to join a collaborative session and operate as a PPD can be discovered by using the Bluetooth device discovery mechanism. This relaxes the constraint to have a specific infrastructure for wireless networking which is still often unavailable in the field as mentioned in section 3. The PPDs will use a peer-to-peer network topology and form a true distributed system this way. This implies there is no central server required to start a collaborative meeting, which is one constraint that needs to be full-filled if an ad-hoc collaborative session is set up. The data contained in the shared information space is distributed among the displays, and coordination is distributed among the peers. This makes the system very flexible for new peers that are joining the collaborative meetings or peers that leave the meeting.

When one of the users wants to edit an object on the shared canvas, his/her device will ask permission to all other PPDs (explained in detail in section 5). Only when the device receives approval from all other displays will the user be allowed to manipulate the object. When the user is done working with the shared object, the PPD will send all changes made by the user to the other PPDs before it releases the mutex lock. This way all data in the shared information space will remain consistent over all PPDs.

5. DISTRIBUTED LOCKING

All the objects used in the applications are shared between

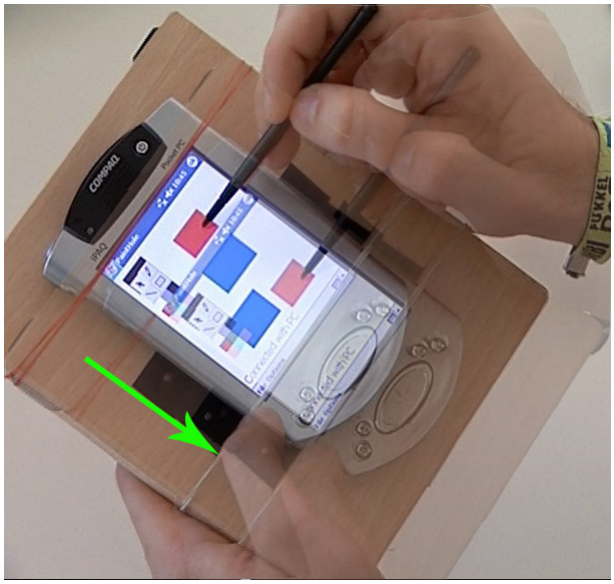

Figure 4: When the user holds a shared object with his or her stylus and moves the PPD at the same time, the object's position will be fixed with respect to the display's position. This enables the user to easily move the object distances greater than the width or height of the display.

the different devices. Therefore we need some kind of control mechanism to ensure that several users do not manipulate the same object at the same moment. This could lead to an inconsistency in the shared information space. Such an inconsistency would be reflected in a different presentation of the same data on devices involved in a collaborative session. Although the users are co-located and verbal communication can be used to coordinate the user's actions on the shared canvas, this could still result in an inconsistent state of the distributed application. Especially objects that appear on multiple screens at the same time are subject to the automatic locking policy. We want to avoid the need for explicit locking – though it is also supported as explained in section 6 – and lower the amount of manual coordination necessary to collaborate on the same surface with two peephole displays.

All displays contain their own version of the shared data. Consistency between the different versions and consequently the different displays has to be guaranteed during the whole collaborative session. To achieve this we use a distributed locking algorithm developed by Ricart and Agrawala [20, 9]. The algorithm uses Lamport clocks [17] since there is no central clock in a distributed system that can be used for synchronization purposes. The locking and unlocking of critical sections works with unique IDs. It is the responsibility of the software developer to assign IDs to all critical sections in the application and to lock the sections before entering them.

In the case of our program, every shared object on the canvas can be treated as a critical section that requires locking. Manipulation of an object is restricted to the display that has access to the critical section of that object. Using the Ricart-Agrawala algorithm we can make sure shared

objects stay in a consistent state when being manipulated by several users. Changes in the state of an object resulting from manipulations by one user will be reflected on the other displays before releasing the lock on the object. This way all users have received the changes before they can edit the shared object themselves.

A PPD can gain access to a critical section by asking permission to all other participating displays as displayed in listing 1. Only when all other displays have given permission to access the shared object(s) the critical section can be entered, as shown in listing 2. When one of the other displays is already in the same critical section, this display will not give permission until the user has finished interacting with the object.

Listing 1: Request access to a critical section

```
WANT_MUTEX[id] := true;
Send request with timestamp T to all other PPDs;

Wait until (number of replies received == number of
     other PPDs)

HAS_MUTEX[id] := true;
WANT_MUTEX[id] := false;
```

Listing 2: Reply to critical section request

```
When ((WANT_MUTEX[id] == false) and (HAS_MUTEX[id] ==
     false))
Then
       Send reply to allow the request;
Else if (WANT_MUTEX[id] == true)
Then
       When (The senders timestamp Ts < my timestamp T)
       Then
              Send reply to allow the request;
       Else if ((Ts == T) and (the senders ID < my ID))
       Then
              Send reply to allow the request;
       Else
              Add the request to request queue;
Else if (HAS_MUTEX[id] == true)
Then
       Add the request to request queue;
```

The Ricart and Agrawala algorithm relies on a multicast-based communication protocol. The algorithm can be improved by using hardware-supported multicast or by only requiring a subset of the votes to lock a critical section. The algorithm is efficient enough to support a limited number of PPDs (up to 4) that are involved in a co-located collaborative session, but we expect for a higher number of PPDs further optimiziations are necessary. In our implementation, we work with shared objects on the information space (section 4.1). Every object has its own ID, which is unique across all PPDs. Therefore the easiest solution is to use the object's ID as distributed mutex ID. Every time a user manipulates a shared object, the distributed mutex with that object's ID will be locked. This way it cannot happen that several PPDs manipulate the same shared object at once. When an object is locked or unlocked, the object is animated to indicate the state change to the other users who have the same object in their scope. Typically, the animation depends on the application type and type of information presented on the shared canvas.

6. EXAMPLE APPLICATIONS

6.1 GeoPlanner

GeoPlanner is a distributed program that can run on two PPDs simultaneously. When the PPDs are connected, their view on the shared information space will be aligned. This way they will see the same part of the shared canvas when their PPDs are at the same physical position. Once the application is running, users can share images and drawings with each other. Both the images and the drawings are shared objects, distributed among the displays. When the objects are added to the shared information space, users can view and manipulate them (figure 5). An advanced distributed locking mechanism (section 5) is used to ensure the consistency of the shared information space.

Figure 5 shows two PPDs running the GeoPlanner application. One of the users added an image, containing a map of Europe to the shared information space. The image has a fixed position with respect to the physical world. The users can see different parts of the map by moving their PPD to the desired position (figure 6). The position of the PPDs is tracked with the V-scope button attached at the top of the PPDs.

Because semantic zooming is supported by the Piccolo library, on which our library is built, all applications using *pizu* support semantic zooming (so does GeoPlanner). Especially when working with large shared information spaces, the ability to zoom can be very helpful. By zooming out, the user will be able to get an overview of the entire workspace. Moving the display up or down causes it to zoom in or out.

There is however a problem that arises when zooming is used with PPDs. An important point of PPDs is that users will see the same view when their PPDs are at the same physical position. When we add zooming to the application, this could result in a different behavior. Suppose one user zooms out, moves a certain distance and zooms in again. Next, the other user moves to the same place the first user holds his/her PPD now, but without the zooming. Both users will see a different part of the image. This happens because the relative distance the view of the PPD moves, depends on the zoom level. When zoomed out, one can cover larger distances on the image with a small movement of the PPD. To solve this problem we adapted the scroll speed of the view to the zoom level. The scroll speed is slowed down when the view is zoomed out, and accelerated when zoomed in. This way, it is possible to combine the advantages of zooming (overview of workspace) and the advantages of peephole displays (shared objects have a fixed position in physical space).

PPDs are very well suited for two-handed interaction. Several studies show that the non-dominant hand is preferred for non-precision tasks, like navigation [8, 15, 16]. Because the PPDs use their position in physical space for navigation, one-handed navigation is possible. Most users will want to use their non-dominant hand for this purpose, leaving the dominant hand free to manipulate the visible objects. The dominant hand will always use the position of the non-dominant hand as a reference [12]. Therefore it is very easy for the user to manipulate data with the dominant hand, while the non-dominant hand moves the display to navigate. Even without watching, a person knows the position of their hands [1], and thus can easily find the display in his non-dominant hand with his other hand.

There are several manipulative actions possible on a shared object. Users can move the object by dragging it with the

Figure 5: Two portable personal displays running the GeoPlanner application. The V-scope sensors attached on top of the PPDs are used to track them in physical space. A map of Europe was added to the shared information space. Both users can move their PPD to see other places on the map. It is also possible to zoom by moving the PPD up or down. The drawing palette on the left side of the screen offers the possibility to make annotations to the image.

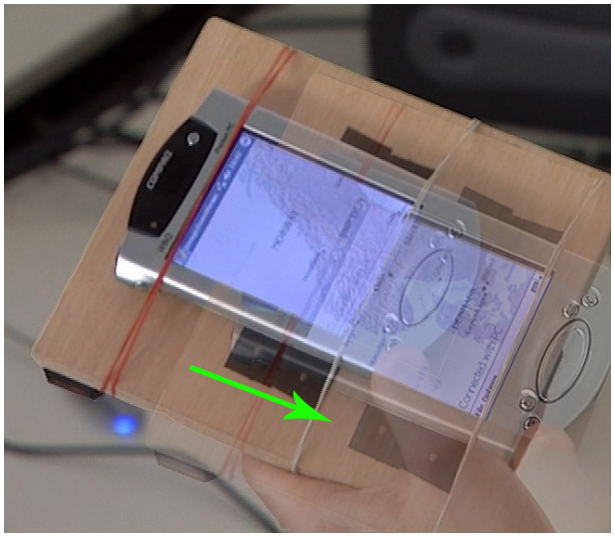

Figure 6: The user can view different parts of the shared information space by moving the PPD to the desired position.

stylus or by holding it with the stylus and moving the display. Besides moving a shared object, it is also possible to manually lock it. When a user locks an object, it cannot be moved or removed without unlocking it first. When manually locked, a little figure of a lock will show up in the top left corner of the shared object. If an object is protected from simultaneous access (described in section 5), an animation of the object notifies the other users who have the same object in their scope.

As explained in section 4.1, every shared object can have other shared objects as its children. These children will move when the parent is moved and will be erased when the parent is erased. A newly created object will automatically become a child of an existing object when it is created on top of that existing object. This is very helpfull when annotating an image on the shared information space (e.g. the map of Europe). All annotations will automatically become children of the image. If a user later on moves the map to another location, the annotations will remain at the correct position on the map.

The idea of PPDs is to enable ad-hoc collaboration among multiple users. Therefore, it will happen at times that some users want to view and/or manipulate the same data. Because the physical position of the display is used to navigate, this can be a cumbersome operation. All users will need to hold their display at the same physical position to view and manipulate the same data, which is not very practical. For this purpose, we offer users the possibility to lock their view. When locked, their view on the shared canvas will at all times remain focused at the locked position, no matter where their PPD is moved to. This way all users can view and/or manipulate the same area on the shared canvas without disturbing eachother. Every change a user makes is immediately visible on all other PPDs, so the users can interact and cooperate on the same data. When done working at the same area, it is possible to realign all displays so all users will see the same data at the same physical position again.

6.2 Us-Draw-It

Us-Draw-It is a distributed drawing program for PPDs (figure 7). The underlying technology is the same as GeoPlanner, it is also built upon the *pizu* PPD library. This application allows users to draw objects (e.g. lines, rectangles, free draw, ...) on a shared information space. The user doesn't need to hold the display still while drawing. Because of this, it is very easy to draw objects many times larger than the PPD's screen. All objects drawn on the canvas are added and updated while they are being drawn. Every user can in real-time see the changes other users make, which enables intuitive ad-hoc collaboration between users.

Us-Draw-It has a number of features in common with GeoPlanner, because they both use the *pizu* library. Similar to GeoPlanner, it is possible to move, erase or lock all drawn objects on the shared information space. Every object drawn on top of an already existing object becomes a child of the existing object. This way it is possible to draw large objects, consisting of many smaller building blocks. In that case, moving or erasing the top parent object will have the same effect on all of its children. Like in GeoPlanner, users can also use zoom in this application. Moving the PPD in the vertical direction causes the application to zoom in or out. Furthermore, locking the PPD's view is also allowed. When locked, the view will not change, wherever the display is moved to and the reference point for tracking will

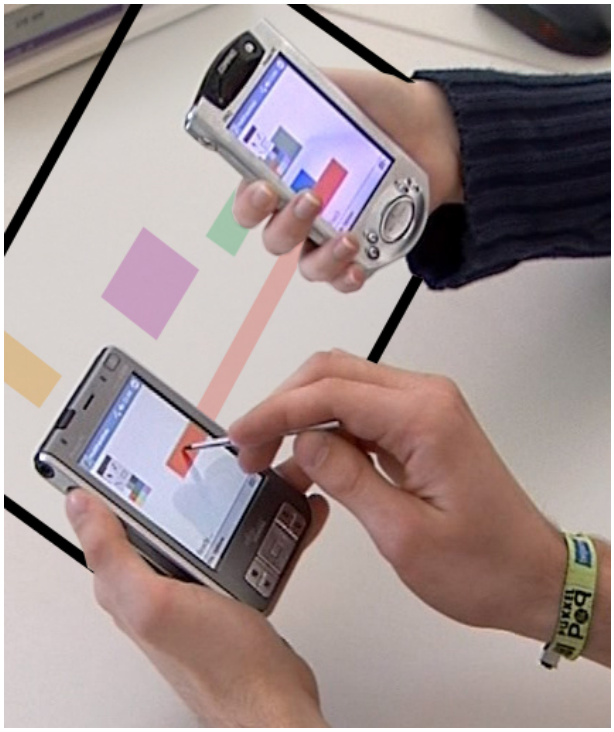

Figure 7: Us-Draw-It is a distributed drawing program for portable personal displays. Every change on one PPD is immediately visible on the other PPD (e.g. moving the red rectangle). This encourages ad-hoc collaboration among the users. The figure shows the information space (semi-transparant) that is spread over the physical space.

be moved together with the display. This allows users to work at the same virtual position on the shared information space without disturbing eachother by holding the PPDs at the same physical position.

7. DISCUSSION

Both example applications were tested informally with a group of four test participants. The participants were subdivided in two separate groups of two people. With "Us-Draw-It", each group had to make one integrated drawing (a drawing of a house) and change some aspects of the drawing afterward. The other test used the "GeoPlanner" application and asked the participant to sort a set of images in a collaborative fashion.

We observed how well the participants could collaborate with the system and how simultaneous interaction is supported. Because of the co-located interaction there is an interesting interaction between coordination by means of verbal communication, automatic coordination (distributed locking) and manual coordination (requesting a lock through the user interface). Despite the possibility to use verbal communication, both tests showed that users often try to manipulate the same object simultaneously and automatic distributed locking is required to preserve a consistent state. We identified two different cases during the tests: one where the virtual space can be shared among the peepholes, and another where this is not possible. The latter is useful for applications that require an efficient locking mechanism: since peepholes can not share the same physical location this also means they can not show the same virtual area therefor this area is only accessible for one user at a time.

However, the first test showed that different users also often need to manipulate the same area of the virtual canvas: although both PDA's are using a different physical position, the same area on the canvas should be shown. For each PDA, the virtual canvas can therefor be translated according to the PDA's position so the different peepholes can show the same area, but the navigation and interaction with the canvas does not change. This also increased efficiency (the test took less time to complete) since there was a better mutual understanding of what the different participants were exactly doing. The automatic locking algorithm proved to be very useful in this situation since it often happened that different users tried to manipulate the same object without manual or verbal coordination beforehand. The 3D navigation possibilities also proved to be an intuitive way to overcome the limited screen space, which confirms the results of [26].

One of the most important drawbacks of the system is its lack of speed because of the delay caused by the tracking technology. We are optimizing different software components of the system to reduce this delay: the network traffic, the graphical user interface and the delay due to the tracking technology. For the graphical user interface we are making the trade-off between the functionality offered by the current toolkit we are using or the speed of an alternative toolkit. A better wireless tracking system (e.g. video tracking) can improve the usability of the example applications a great deal.

Since we did the evaluation inside of the lab (due to the lack of portability of current tracking technologies), it is difficult to make assumptions about the usability of the system in its target context: ad-hoc meetings "on the spot" with only a mobile device that can be used by each participant. We plan to extend the system so it can be used in a broader context. This would allow us to conduct more extensive field testing .

8. CONCLUSIONS

In this paper we introduced a system to support collaboration in mobile co-located meetings. With an increasing amount of mobile devices being used our system offers new possibilities to cope with a limited screen size, support collaboration and offer new ways to visualize and collaboratively interact with large information spaces. Based on the ideas of the peephole displays, which was implemented for individual usage on mobile devices by Ka-Ping Yee, our solution is simple and intuitive. Tests have shown that although a location-aware display is not necessarily faster or more efficient for applications that can be shown within the limited screen space, in a constrained environment they are more intuitive and usable. In our implementation we extended the Piccolo library and added support for multiple synchronized input/output peepholes. This results in a library that can be used to build custom applications suitable for collaborative co-located interaction using multiple devices. Although 3D tracking technology which is precise enough for our goals is not sufficiently compact and mobile yet, our experiments show this is the only factor that prevents the applications

presented in this paper from being used in realistic mobile settings. It does provide a usable framework to research interaction techniques that can be used for co-located collaboration with multiple mobile devices, an area that is becoming increasingly important but has not been thoroughly explored yet.

Acknowledgments

Part of the research at EDM is funded by EFRO (European Fund for Regional Development), the Flemish Government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT). Funding for this research was also provided by the Research Foundation – Flanders (F.W.O. Vlaanderen, project number G.0461.05).

9. REFERENCES

[1] Ravin Balakrishnan and Ken Hinckley. The role of kinesthetic reference frames in two-handed input performance. In *ACM Symposium on User Interface Software and Technology*, pages 171–178, 1999.

[2] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer. Toolkit design for interactive structured graphics. 2003.

[3] Benjamin B. Bederson, Jon Meyer, and Lance Good. Jazz: an extensible zoomable user interface graphics toolkit in java. In *UIST*, pages 171–180, 2000.

[4] Devesh K Bhatnagar. Position trackers for head mounted display systems: A survey. Technical report, University of North Carolina, Chapel Hill, NC, USA, 1993.

[5] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and Magic Lenses: The See-Through Interface. *Computer Graphics*, 27:73–80, 1993.

[6] Official bluetooth site. World Wide Web. `http://www.bluetooth.com/`.

[7] Andreas Butz and Antonio Krüger. A Generalized Peephole Metaphor for Augmented Reality and Instrumented Environments. In *Proceedings of The International Workshop on Software Technology for Augmented Reality Systems (STARS)*, 2003.

[8] W. Buxton and B. Myers. A study in two-handed input. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–326, 1986.

[9] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems (3rd ed.): concepts and design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[10] George W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Commun. ACM*, 36(7):39–49, 1993.

[11] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, Massachusetts, 1994.

[12] Yves Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *The Journal of Motor Behavior*, 19(4):486–517, 1987.

[13] Kasper Hornbk, Benjamin B. Bederson, and Catherine Plaisant. Navigation patterns and usability of overview + detail and zoomable user interfaces for maps. Technical report, University of Maryland, Human-Computer Interaction Lab, College Park, MD, USA, nov 2001.

[14] Kasper Hornbk, Benjamin B. Bederson, and Catherine Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Trans. Comput.-Hum. Interact.*, 9(4):362–389, 2002.

[15] Paul Kabbash, William Buxton, and Abigail Sellen. Two-handed input in a compound task. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 417–423, 1994.

[16] Paul Kabbash, I. Scott MacKenzie, and William Buxton. Human performance using computer input devices in the preferred and non-preferred hands. In *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 474–481, 1993.

[17] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[18] Piccolo toolkit. World Wide Web. `http://www.cs.umd.edu/hcil/piccolo/learn/`.

[19] A. Lastra R. Holloway. Virtual environments: A survey of the technology. In *SIGGRAPH'95 Course*, pages A.1–A.40, 1995.

[20] Glenn Ricart and Ashok K. Agrawala. An optimal algorithm for mutual exclusion in computer networks. *Commun. ACM*, 24(1):9–17, 1981.

[21] Baillot Y. Rolland, P.J. and Goon A. A survey of tracking technology for virtual environments. Technical report, University of Central Florida, Center for research and education in optics lasers (CREOL), Orlando FL 32816, 1999.

[22] Robert J. Stone. Position and orientation sensing in virtual environments. *Sensor Review*, 16(1):40–46, 1996.

[23] Michael Tsang and George W. Fitzmaurice et al. Boom Chameleon: Simultaneous Capture of 3D Viewpoint, Voice and Gesture Annotations on a Spatially-aware Display. In *UIST '02*, pages 111–120, 2002.

[24] Greg Welch and Eric Foxlin. Motion tracking: No silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–38, 2002.

[25] Andrew Wilson and Steven Shafer. Xwand: Ui for intelligent spaces. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 545–552, 2003.

[26] Ka-Ping Yee. Peephole Displays: Pen Interaction on Spatially Aware Handheld Computers. In *Conference on Human factors in computing systems*, pages 1–8, 2003.