T3: Rapid Prototyping of High-Resolution and Mixed-Presence Tabletop Applications

Philip Tuddenham and Peter Robinson
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge CB3 0FD, UK
{firstname}.{lastname}@cl.cam.ac.uk

Abstract

Multi-person tabletop applications that require a high display resolution, such as collaborative web-browsing, are currently very difficult to create. Tabletop applications that support mixed-presence collaboration, whereby some collaborators are remote, are also hard to build. As a consequence, there has been little investigation of important tabletop applications, despite promising early results. In this paper, we present T3, a software toolkit that addresses these challenges. T3 allows researchers to rapidly create high-resolution multi-person tabletop applications for co-located or remote collaborators. It uses multiple projectors to create a single seamless high-resolution tabletop display, and allows multiple tabletops to be connected together to support mixed-presence collaboration. This engineering is hidden behind a simple, flexible API that can be used to implement the vast majority of today's tabletop applications. T3 also supports existing user interface components, including buttons, web-browsers and spreadsheets, allowing the rapid creation of complex tabletop applications. We show how we have used T3 to create five novel tabletop applications that would previously have been very difficult to build.

1. Introduction

In recent years, interactive tabletop interfaces have emerged as a key tool for co-located collaboration over digital artifacts. Yet, in spite of much promising research, there has been little investigation of tabletop interfaces to support the collaborative tasks for which people currently use their desktop computers, such as collaborative web browsing, spreadsheets and document review. These are compelling applications to which tabletop interfaces may bring significant benefits. However, this area remains largely unexplored because, with very few exceptions, the display resolution of today's tabletop interfaces is too low to support such applications.

A further area that remains largely unexplored is the extension of tabletop interfaces to remote groups of collaborators. In this mixed-presence setting, each remote group would sit at their own tabletop. As shown in Figure 1, all the tabletops would then be linked together to provide a shared workspace for collaboration in which collaborators could interact with, position and orientate digital artifacts. All the tabletops would show the same artifacts, along with remote embodiments (such as arm shadows) of the participants. Such a system could well offer remote collaborators some of the benefits of tabletop interaction, such as a greater awareness of each others' actions, and space to explore both personal and group work [13], both longstanding problems in conventional groupware. However, despite promising early results, investigation of these mixed-presence systems has been rather limited [16, 7, 3].

The significance of these two gaps in the research should not be underestimated. If we are truly to believe that these interfaces will be adopted then we must begin to explore real-world tasks, and the benefits that tabletop interfaces can offer over and above conventional physical tabletops.

Recent tabletop research has been fuelled by software toolkits that address the core engineering involved, allowing researchers to concentrate on interaction techniques and applications. As we shall show, the reason for these two gaps in the research is that these kinds of systems pose unique engineering challenges that cannot easily be solved using today's toolkits. From our own experiences, and from discussions at a workshop [17], we know that it is presently very difficult for tabletop researchers to overcome these issues.

In this paper we present T3, a software toolkit that we have implemented to address these problems. It uses multiple projectors to create a single seamless high-resolution tabletop display, and allows multiple tabletops to be connected together. T3 allows researchers to rapidly prototype high-resolution tabletop interfaces for applications such as collaborative spreadsheets and collaborative web-browsing, both for co-located and mixed-presence collaboration. It is freely available for academic research and will allow rapid exploration of this field. Furthermore, through our experiences, we have been able to investigate the challenges in creating these interfaces.

In the next section, we review recent work to identify the challenges in engineering these applications and establish design goals for our work. We then present T3, showing how it meets these goals, and outline a simple worked example. We illustrate its utility through five novel research projects, discuss its limitations and conclude with recommendations for further exploration of the field.

2. Background and Design Goals

2.1. Tabletop Collaboration

A great number of projects have investigated various aspects of tabletop interfaces for co-located collaboration [e.g. 15]. Much of the work has been possible because of reusable software toolkits, notably DiamondSpin [14], and the more recent Buffer Framework [5], that handle the core engineering such as:

- Allowing collaborators to arbitrarily position and orient digital artifacts, and groups of digital artifacts.
- Allowing multiple users to interact concurrently.
- Supporting direct interaction using bare-hands or stylus.

DiamondSpin is particularly useful because it allows researchers to rapidly prototype complex tabletop applications by reusing existing Java Swing user interface components, such as buttons and file choosers. This feature is essential if we are to investigate complex applications such as collaborative spreadsheets or web-browsing, because it is

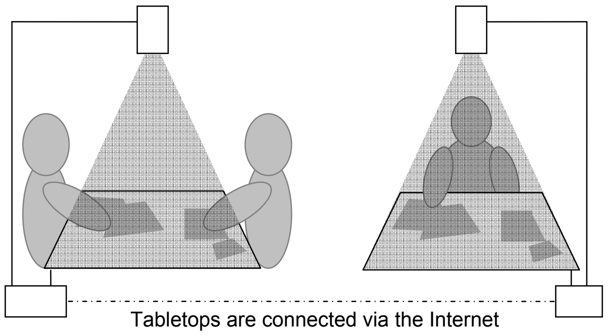

Figure 1: Mixed-presence collaboration. The tabletop provides a shared visual workspace both for the co-located participants and the remote participants.

not within the scope of a research project to engineer such applications from scratch.

By contrast, the Buffer Framework provides good performance when hundreds or thousands of digital artifacts appear on the tabletop, but does not allow the reuse of existing user interface components, and so creating new applications is difficult. To our knowledge, it has thus far been used only to create an application for moving and grouping photos.

2.2. Higher-Resolution Displays

The key problem in supporting applications such as collaborative web-browsing or spreadsheets, is in creating a display surface with a sufficiently high resolution. In order to use the unique affordances of tabletop collaboration, we wish each web-page or spreadsheet to appear no larger than an ordinary sheet of paper, so that they can be passed between collaborators as one might with paper documents. However, to accomplish this, we need to be able to legibly display small text and user interface components, which is impossible on most of today's tabletop displays.

We aimed to display 12pt text legibly (i.e. text that appears the same size as 12pt text that comes out of the printer). We have found that this requires a resolution of at least 60dpi, so that a fairly modest 85cm $\times$ 85cm table requires a 4 Megapixel display. By contrast, almost all today's tabletop displays provide at most 2.6 Megapixels using at most two projectors.

Higher-resolution projectors are extremely expensive and unsuitable. By far the easiest way to create a higher-resolution display using is to tile multiple projectors. For example, we have tiled 6 projectors to create a 4.7 Megapixel display using modest, inexpensive equipment. Such multi-projector designs are commonplace in the display walls used in visualisation research. However, they introduce further problems. Firstly, the huge number of pixels can lead to unresponsive applications. Secondly, it is impossible to align the projectors to the degree of precision required to make a perfect seamless display, and so these displays can suffer from small overlaps, mismatches and keystoning. To compensate, small adjustment transformations and blending masks must be applied to each frame before it is sent to the projectors.These solutions are well-known and employed in the software toolkits that are widely used to create large multi-projector display walls [9, 20, 18, 1]. However, these toolkits are not designed to afford tabletop interaction or rapid prototyping; they typically do not, for example, support rotation of artifacts.

The DiamondSpin toolkit cannot easily be extended to a multi-projector tiled display because of its rendering mechanism, whereas the Buffer Framework performed well on a 4 projector tiled display.

2.3. Mixed-Presence Collaboration

We also wish to investigate mixed-presence collaboration, whereby two geographically-separated tabletops are linked together to allow two remote groups to collaborate as though co-located around the same tabletop. Prior research in this area has investigated mixed-presence drawing surfaces [e.g. 16] and tangible interfaces [e.g. 2, 19].

There has, however, been little mixed-presence investigation of applications in which collaborators interact with, position and orientate digital artifacts like spreadsheets or text documents. TIDL [7] and RemoteDT [3] both allow mixed-presence collaborators to position and interact with digital artifacts on large horizontal displays (much as GroupKit [10] does for remote collaborators on conventional desktop computer interfaces). However, none of these systems allow collaborators to reorient artifacts. Orientation serves several important roles in co-located tabletop collaboration, such as allowing transitions between personal and group work [13, 8], and it should not be overlooked when designing mixed-presence tabletop systems.

Creating a mixed presence system that allows participants to reorient artifacts is desirable but technically difficult: firstly because standard remote display protocols are not designed to handle artifact rotation; and also because the existing tabletop toolkits, DiamondSpin and Buffer Framework, cannot easily be extended to support mixed-presence collaboration.

2.4. Design Goals

Having reviewed the literature, we now establish several design goals. The system should:

- provide abstractions to support the core tabletop interaction functionality provided by other tabletop toolkits, such as rotation of artifacts. Such requirements have been discussed in prior work [11, 14], and in Section 2.1 we have enumerated the most salient given the space available.
- allow creation of higher-resolution tabletops by supporting multiple projectors in a tiled array, applying small transformations and masks to create the illusion of a seamless display.
- allow geographically-separated tabletops to be connected together to create a shared workspace for investigation of mixed-presence applications, in which collaborators interact with, position and orientate digital artifacts.
- allow researchers to rapidly create complex tabletop applications like spreadsheets by reusing existing user interface components.

Table 1 summarises our design goals and the limitations of existing tools.

	Diamond Spin	Buffer F'work	Display walls	TIDL/ RemoteDT	T3
Tabletop interaction (e.g. rotation, etc.)	✓	✓			✓
Higher-resolution		✓	✓		✓
Two linked surfaces (mixed-presence)			Some	✓	✓
Reuse existing UI components	✓		✓	✓	✓

Table 1: Comparing tools and design goals.

3. The T3 Toolkit

3.1. Overview

Having established these design goals, we now provide a brief overview of the T3 architecture, followed by a description of each component, example applications, and a discussion of the implementation.

Figure 2 outlines the T3 architecture. Applications are created using a simple, well-documented Java API. The API is based around the notion of a single seamless large display, in which the application programmer can create rectangular tiles to represent interactive draggable digital artifacts such as spreadsheets or web-pages. The application programmer uses T3 to arbitrarily position and orient the tiles within a single large coordinate space.

Importantly, tiles can be filled with existing Java Swing user interface components, which function as expected without any extra code required on the part of the applications programmer. This includes buttons and text boxes, and even third-party components such as spreadsheets and web-browsers, allowing rapid creation of complex applications. T3 works behind the scenes to ensure that the application programmer is never required to consider the effects of tile rotation or scale, nor how to distribute the information to multiple tabletops, nor multi-projector blending techniques, nor simultaneous input event streams.

Each multi-projector tabletop display is controlled by a local computer running a T3 client. This client communictes with the application, receiving tile updates and sending back user input events. If desired, multiple clients can connect to the same application via the Internet to allow mixed-presence collaboration, as proposed in the Introduction. The client automatically creates the correct images to send to each of the projectors in the multi-projector display. It positions and orients the tiles, applying small warps and masks to correct for projector misalignment and create the illusion of a single seamless display.

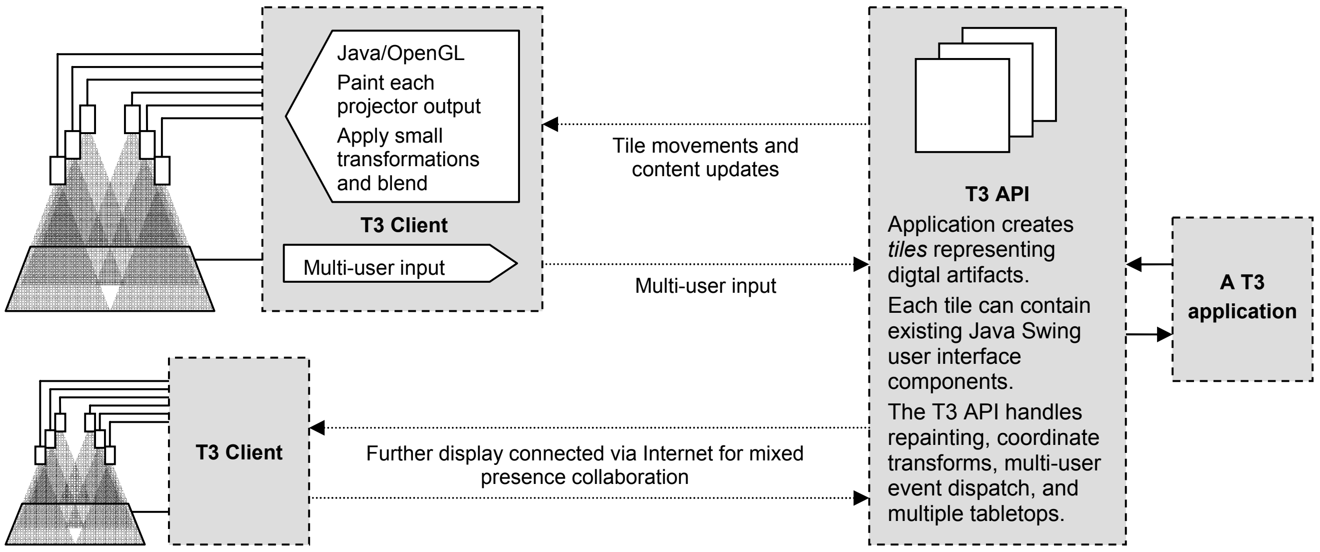
Java/OpenGL
Paint each projector output
Apply small transformations and blend
T3 Client
Multi-user input
Tile movements and content updates
Multi-user input
T3 API
Application creates *tiles* representing digtal artifacts.
Each tile can contain existing Java Swing user interface components.
The T3 API handles repainting, coordinate transforms, multi-user event dispatch, and multiple tabletops.
A T3 application
T3 Client
Further display connected via Internet for mixed presence collaboration

Figure 2: T3 system architecture to support multi-projector tabletops and mixed-presence collaboration.

3.2. Physical Apparatus

T3 is easily configurable to support any number of projectors connected to a single PC. We have created three tabletop displays:

- Six projectors and a single PC with three dual-head graphics cards to create a 4.7 Megapixel display.
- Four projectors and a single PC with two dual-head graphics cards to create a 5.7 Megapixel display.
- One projector to create a 0.8 Megapixel display.

All the components are available off-the-shelf, and neither multi-projector display cost more than $13,000, including mountings. We have tested the toolkit using 3 brands of graphics cards. For multi-user input, we use standard graphics tablet styluses and Anoto streaming styluses, and the system will also extend to multi-touch surfaces.

3.3. API and Swing Applications

The application programmer creates tiles by instantiating T3's tile class. The programmer specifies how each tile should be positioned, rotated and scaled on the tabletop by specifying coordinates and dimensions in millimetres, and an angle in radians. Collaborators move tiles around the display surface by dragging with their stylus (or their finger, if a multi-touch surface is used) using Rotate 'N' Translate [8]. The application programmer determines which tiles are draggable and can group tiles so that they are then dragged together, allowing creation of mobile container elements like Storage Bins [12].

Each tile functions as a Java Swing window in which existing user interface components can be used without any modification required, as we illustrate later with a worked example (Section 4). Multiple collaborators can interact simultaneously to drag these tiles and manipulate the components within, using stylus or bare hand input, depending on the hardware available. The vast majority of Swing components work as expected without any modification, though components that use popup windows currently require special attention.

This mechanism works by opening the necessary Swing windows off-screen. Swing repaint events are then trapped by the T3 toolkit which renders the windows into images that are sent to the T3 clients. Similarly, user input events received from the T3 clients are translated into Swing input events and dispatched to the appropriate Swing window. T3 uses geometric transforms to determine the tile immediately "underneath" the event on the tabletop and then transforms the event from tabletop coordinates (in millimetres) into Swing window coordinates, "undoing" the effects of tile rotation, translation and scaling, similar to DiamondSpin's transformation engine. The resulting Swing event objects are also augmented with extra fields representing the absolute coordinates in millimetres and the person who caused the event so that applications can use this information.

T3 also provides an alternative API to support more customised applications that avoid the constraints of Java Swing. This allows implementation of more complex designs like Storage Bins [12] or Interface Currents [6], and both APIs can be mixed within the same application to produce, for example, an Interface Currents design that supports web browsing using a Java Swing web browser component. In this alternative API, the application programmer simply overrides a paint routine and an input event process-

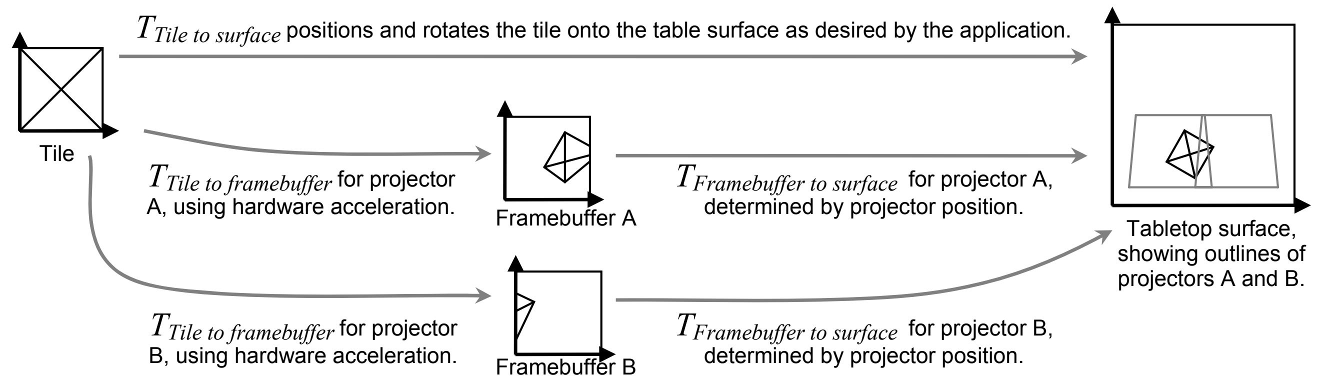
$T_{Tile\ to\ surface}$ positions and rotates the tile onto the table surface as desired by the application.
Tile
$T_{Tile\ to\ framebuffer}$ for projector A, using hardware acceleration.
Framebuffer A
$T_{Framebuffer\ to\ surface}$ for projector A, determined by projector position.
Tabletop surface, showing outlines of projectors A and B.
$T_{Tile\ to\ framebuffer}$ for projector B, using hardware acceleration.
Framebuffer B
$T_{Framebuffer\ to\ surface}$ for projector B, determined by projector position.

Figure 3: Transformations between coordinate spaces in the T3 client.

ing routine for each tile.

T3 uses a multi-threaded architecture to handle the simultaneous input event streams from multiple users and multiple tabletops. This provides responsive performance and allows multiple collaborators to interact simultaneously to drag or manipulate tiles. The multi-threading and object locking is handled automatically by the toolkit.

Tiles are rectangular by default, but other shapes can be created by painting parts of the tile with transparent pixels. An ordering for the tiles determines occlusion when tiles overlap. T3 also provides a mechanism to smoothly animate groups of tiles between two specified positions and orientations, which can be used, for example, to zoom out to a thumbnail overview. Again, T3 handles the multi-threading and object locking. A further mechanism allows the creation of translucent lines that join different tiles, which can be used, for example, to illustrate dependencies between artifacts.

3.4. Display Management

Each multi-projector display is controlled by a local T3 client, which receives tile information from the application, via the Internet if necessary, and performs the actual rendering. The client uses OpenGL (via the JOGL library) to exploit hardware-accelerated rendering provided by modern graphics cards and, for the applications described in this paper, we achieved a frame rate of 60fps at all times, using fairly modest graphics hardware.

Tile images are stored in the texture memory within each graphics card. The client then renders each frame for each projector by transforming tile images into the framebuffer. Figure 3 illustrates the transformations between the tile coordinate space, the frame buffer, and the display surface. The transformation, $T_{Tile\ to\ framebuffer}$ consists of two parts:

$$T_{Tile\ to\ framebuffer} = T^{-1}_{Framebuffer\ to\ surface} \cdot T_{Tile\ to\ surface}$$

The first part, $T_{Tile\ to\ surface}$, positions, scales and rotates the tile onto the display surface as desired by the application. The second part, $T^{-1}_{Framebuffer\ to\ surface}$, is a small transformation that compensates for projector misalignment. It is obtained using a short calibration procedure, which calculates the relationship between points in the framebuffer and points on the display surface, and then performs a matrix inversion. As described previously, the misalignment problem is virtually impossible to solve mechanically for high-resolution displays, and such software solutions are well understood in the display-wall community.

3.5. Multiple Tabletops

The client and the application can run on the same computer to create a single high-resolution multi-projector tabletop interface. Alternatively, as described earlier, multiple clients can connect to the application via the Internet, to investigate mixed-presence tabletop collaboration. In this case, the clients and the server use a protocol to communicate tile content updates, tile movement and user input events.

This protocol is optimised so that the most frequent operations, such as dragging tiles, use low bandwidth and provide reponsive performance. The system also uses a basic adaptive scheme to avoid overwhelming lower-performing clients. We have tested our system using both a high speed LAN and a lower bandwidth wireless network, and it performed responsively in both cases.

In mixed-presence collaboration, we display remote embodiments on the tabletops in order to convey presence and to allow remote participants to gesture to each other. We currently use telepointer traces [4] that follow each participant's stylus (or hands, if using a bare-hand multi-touch interaction surface). Traces are a starting point in our investigation of remote embodiments: they are easy to implement, allow rich gestures, and are robust to network jitter.

```
public static void main(String[] a) throws Exception {

// Create a new tabletop, d.
PortfolioServer d = new PortfolioServer(
     new ServerSocket(2000),
     new ServerSocket(2001), false);

// create a new tile of 500*500 pixels on d
SwingFramePortfolio myTile = new SwingFramePortfolio(
     d, d.rootPortfolio, new RotateNTranslate(),
     500,500);

// add a JSpreadsheet and menu bar to the tile
myTile.getFrame().add( new JSpreadsheet(80,40) );
myTile.getFrame().setJMenuBar(menuBar);

// make the tile appear a physical size 200mm*200mm
myTile.setTileWidthAndHeightInPORT(200.0, 200.0);

// position the tile 300mm from the top and left of
// the tabletop, rotate it by 0.17 radians  and set
// scale factor 1.00.
myTile.setPORTtoPPORT(300.0, 300.0, 0.17, 1.00);

// make the tile visible
myTile.setVisibleWhenParentVisible(true);

}
```

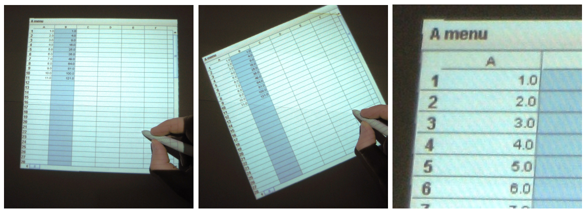

Figure 4: Complete sample code (top) to create a tabletop spreadsheet application (bottom) using T3. The spreadsheet appears on the multi-projector tabletop. It can be passed between collaborators using Rotate 'N' Translate, and is editable.

They allow participants to convey shapes, routes and indicate groupings. However, they may not convey presence as well as alternative embodiments such as arm shadows [16].

4. T3 Applications

We now describe a range of projects that rely on T3 to explore new tabletop applications. The projects are all being undertaken by five students in our Laboratory, and would be very difficult to implement without the core functionality that T3 provides.

Worked Example. Figure 4 illustrates a short program that creates a tile and fills it with a third party Swing JSpreadsheet component. The result is a working tabletop spreadsheet application. The spreadsheet appears as a legible rectangular 20cm × 20cm tile on the table. It is sufficiently small that multiple spreadsheets can be viewed simultaneously by collaborators around the table, and can be passed between collaborators using the Rotate 'N' Translate technique. Columns and rows can be selected, and formulae can be entered into cells. Further development of this basic application will allow participants to collaborate over multiple interdependent worksheets to perform different analyses of the same data set.

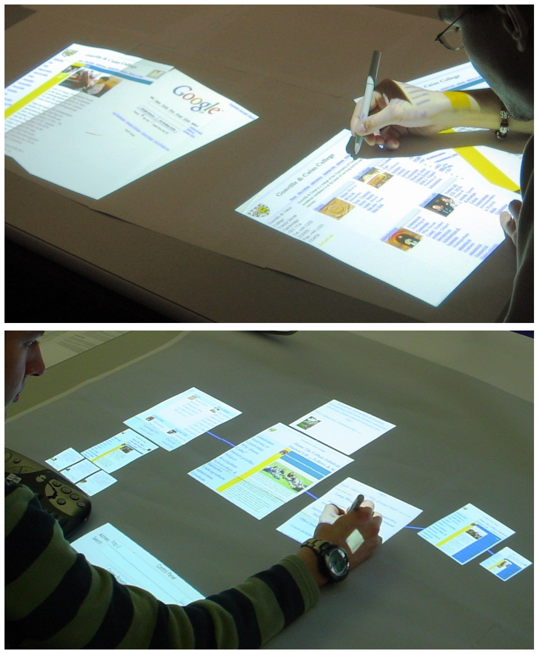

Figure 5: Collaborative web-browsing using T3. Web pages appear small yet legible. They can be passed around the table (top) and browsed in a tree (bottom).

Collaborative Web-Browsing. A second project investigates finding and sharing information from the web (Figure 5). The high-resolution capability of T3 allows web pages to appear legible yet sufficiently small that multiple collaborators can each read their own pages and pass them around. T3 also allows us to reuse a third party Java Swing web browser component for rapid development; the basic tabletop web browsing application was implemented by a student in 60 lines of Java in around 1 hour. Collaborators can open web pages, follow links, and pass pages between each other.

Remote Document Review Meetings. Our third project investigates mixed-presence document review meetings, in which remote collaborators discuss and collaboratively annotate draft text documents. Our interface allows multi-page text documents to appear rather like an open book on the tabletop (Figure 6). We use the high-resolution feature of T3 to project legible text at size 12pt. The interface allows remote collaborators to use styluses to pass documents to each other, to browse within a document using a thumbnail view, and to annotate pages. T3 allowed us to rapidly

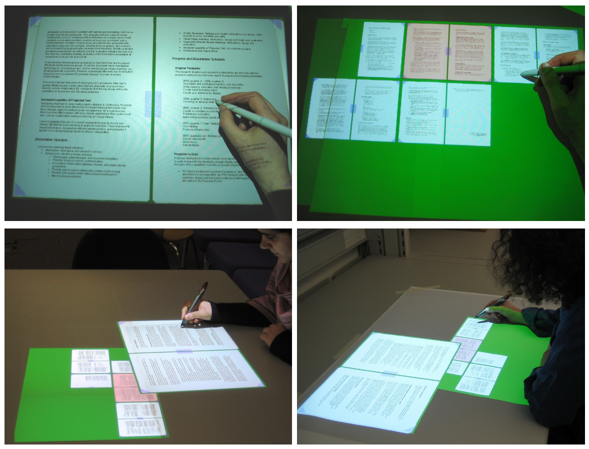

Figure 6: Remote review meetings using T3. Documents containing size 12pt text can be read (top-left), browsed (top-right), and used for remote collaboration (bottom-left and bottom-right).

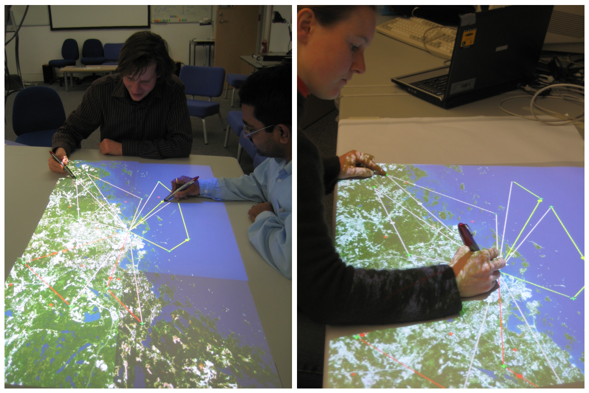

Figure 7: Mixed-presence command and control interfaces using T3. Two co-located participants (left) collaborate with a remote participant (right).

create the interface, which was implemented by a graduate student in 650 lines of Java in around three days.

Command and Control Interfaces. A further student project uses T3 to investigate map-based command and control tasks, comparing co-located collaboration to mixed-presence collaboration (Figure 7). T3 has allowed us to create a single interface and then switch easily between co-located collaboration and mixed-presence collaboration to conduct the study.

Large-Format Collaborative Programming Interfaces. Students have used T3's high-resolution capabilities to generate interactive tabletop UML diagrams annotated with source code (Figure 8).

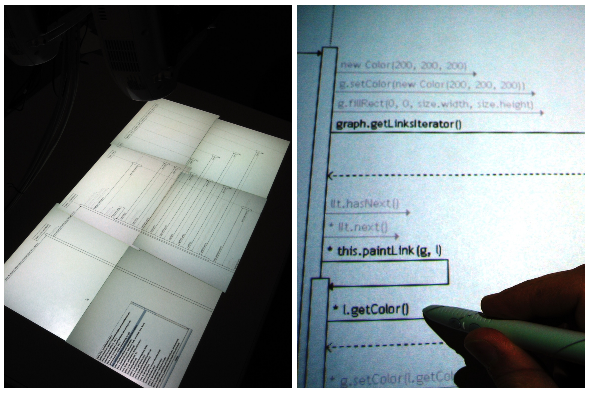

Figure 8: Interactive UML sequence diagrams with annotations using T3.

5. Discussion

T3 addresses the core engineering required to use multi-projector displays and to connect remote tabletops together. However, it nevertheless provides a simple, flexible API that allows researchers to rapidly prototype new tabletop applications, without having to consider the engineering realities of multi-projector transformations or distributed rendering.

The ability to reuse existing Java Swing user interface components, such as buttons and spreadsheets is not without its limitations. The components are designed only for a single user and so, for example, in our tabletop spreadsheets it is not possible for two users to simultaneously select two different columns within the same spreadsheet. Furthermore, we do not believe that applications designed for a single user at a desktop PC are by themselves sufficient to support tabletop collaboration. Nevertheless, the ability to use existing components will undoubtably lead to rapid development of more complex tabletop applications, such as collaborative web-browsing. T3 also supports more radical application designs by providing an alternative API that is not constrained by Java Swing (Section 3.3).

T3 integrates recent research from co-located tabletop interfaces, high-resolution multi-projector display walls and distributed groupware. The unique combination of high-resolution applications, mixed-presence collaboration, tabletop interaction techniques and rapid prototyping of complex tabletop applications, will further tabletop research in a way that has not previously been possible. For example, the combination of high-resolution, tabletop interaction and complex applications allows rapid prototyping of tabletop web browsing interfaces. Similarly, mixed-presence collaboration and tabletop interaction permits interfaces in which mixed-presence collaborators can reorient artifacts, a behaviour which serves important awareness roles in co-located collaboration.

6. Conclusions

Despite a great deal of successful research into tabletop collaboration, two important areas remain largely unexplored: applications that require higher-resolution displays, such as collaborative web-browsing; and mixed-presence collaboration, whereby tabletops provide a shared visual workspace for geographically-separated collaborators. Our analysis shows that such applications are particularly difficult for researchers to create, because they involve engineering problems that are not addressed by existing tools.

In this paper, we presented the T3 software toolkit, which addresses these problems. T3 allows researchers to rapidly create higher-resolution tabletop applications for both co-located and mixed-presence collaboration. It addresses the engineering required to create seamless multi-projector tabletop displays and to connect multiple tabletops together. T3 presents a simple, flexible API that can be used to implement the vast majority of today's tabletop applications. Existing Java Swing user interface components, including buttons and spreadsheets, can be used without modification, allowing researchers to rapidly create complex applications. We illustrated the utility of T3 with five novel projects that would be infeasible without it.

T3 is freely available to academic researchers, along with documentation and example applications, at `http://www.cl.cam.ac.uk/users/pjt40/`. We hope that it will enable rapid exploration of these areas.

Acknowledgements

We gratefully acknowledge the support of Rowan Hill, Tom Matthews, Richard Russell and Stephen Williams, who are undertaking projects using T3; and Thales Research and Technology (UK) and the EPSRC, who jointly funded this work.

References

[1] Ashdown, M., Flagg, M., Sukthankar, R., and Rehg, J. A flexible projector-camera system for multi-planar displays. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR) 2004*, vol. 2, 165–172.

[2] Brave, S., Ishii, H., and Dahley, A. Tangible Interfaces for Remote Collaboration and Communication In *Proc. CSCW 1998*, 169–178.

[3] Esenther, A., and Ryall, K., RemoteDT: Support for Multi-Site Table Collaboration. In *Proc. Int. Conf. Collaboration Technologies (CollabTech)*, 2006.

[4] Gutwin, C. and Penner, R. Improving interpretation of remote gestures with telepointer traces. In *Proc. CSCW 2002*, 49–57.

[5] Isenberg, T., Miede, A., and Carpendale, S. A buffer framework for supporting responsive interaction in information visualization interfaces. In *Proc. Int. Conf. Creating, Connecting and Collaborating through Computing (C5'06)*, IEEE Computer Society (2006), 262–269.

[6] Hinrichs, U., Carpendale, S., Scott, S.D., and Pattison, E. Interface currents: Supporting fluent collaboration on tabletop displays. In *Proc. Smart Graphics*, 2005.

[7] Hutterer, P., Close, B. S., and Thomas, B. H. Supporting Mixed Presence Groupware in Tabletop Applications In *Proc. TABLETOP'06*, 63–70.

[8] Kruger, R., Carpendale, S., Scott, S.D., and Tang, A. Fluid integration of rotation and translation. In *Proc. CHI 2005*, 601–610.

[9] Li, K., Chen, H., Chen, Y., Clark, D.W., Cook, P., Damianakis, S., Essl, G., Finkelstein, A., Funkhouser, T., Housel, T., Klein, A., Liu, Z., Praun, E., Samanta, R., Shedd, B., Singh, J.P., Tzanetakis, G., and Zheng, J. Building and using a scalable display wall system. *IEEE Comp. Graphics and Applications 20*, 4 (2000), 29–37.

[10] Roseman, M. and Greenberg, S. Building Real Time Groupware with GroupKit, A Groupware Toolkit. *ACM Trans. Comp. Human Interaction 3*, 1 (1996), 66–106.

[11] Scott, S., Grant, K., and Mandryk, R. System guidelines for co-located collaborative work on a tabletop display. In *Proc. ECSCW 2003*, 2003.

[12] Scott, S.D., Carpendale, M.S.T., and Habelski, S. Storage bins: Mobile storage for collaborative tabletop displays. *IEEE Comp. Graphics and Applications 25*, 4 (2005), 58–65.

[13] Scott, S.D., Sheelagh, M., Carpendale, T., and Inkpen, K.M. Territoriality in collaborative tabletop workspaces. In *Proc. CSCW 2004*, 294–303.

[14] Shen, C., Vernier, F.D., Forlines, C., and Ringel, M. Diamondspin: an extensible toolkit for around-the-table interaction. In *Proc. CHI 2004*, 167–174.

[15] Shen, C., Ryall, K., Forlines, C., Esenther, A., Vernier, F.D., Everitt, K., Wu, M., Wigdor, D., Ringel Morris, M., Hancock, M., and Tse, E. Informing the Design of Direct-Touch Tabletops *IEEE Comp. Graphics and Applications 26*, 5 (2006), 36–46.

[16] Tang, A., Boyle, M., and Greenberg, S. Understanding and mitigating display and presence disparity in mixed presence groupware. *J. Research and Practice in Information Technology 37*, 2.

[17] Terrenghi, L., May, R., Baudisch, P., MacKay, W., Paterno, F., Thomas, J., and Billinghurst, M. Information visualization and interaction techniques for collaboration across multiple displays. In *Ext. Abstr. CHI '06*, 1643–1646.

[18] Wallace, G., Anshus, O.J., Bi, P., Chen, H., Chen, Y., Clark, D., Cook, P., Finkelstein, A., Funkhouser, T., Gupta, A., Hibbs, M., Li, K., Liu, Z., Samanta, R., Sukthankar, R., and Troyanskaya, O. Tools and applications for large-scale display walls. *IEEE Comp. Graphics and Applications 25*, 4 (2005), 24–33.

[19] Wilson, A. D., and Robbins, D. C. PlayTogether: Playing Games across Multiple Interactive Tabletops *IUI'07 workshop on Tangible Play*.

[20] Yang, R., Gotz, D., Hensley, J., Towles, H., and Brown, M.S. Pixelflex: a reconfigurable multi-projector display system. In *Proc. Visualization '01*, IEEE Computer Society, Washington, DC, USA (2001), 167–174.