A Concept for Sketchable Workspaces and Workflows

Raphael Wimmer
University of Regensburg
Universitätsstr. 31
93053 Regensburg
Germany
raphael.wimmer@ur.de

Jürgen Hahn
University of Regensburg
Universitätsstr. 31
93053 Regensburg
Germany
juergen.hahn@ur.de

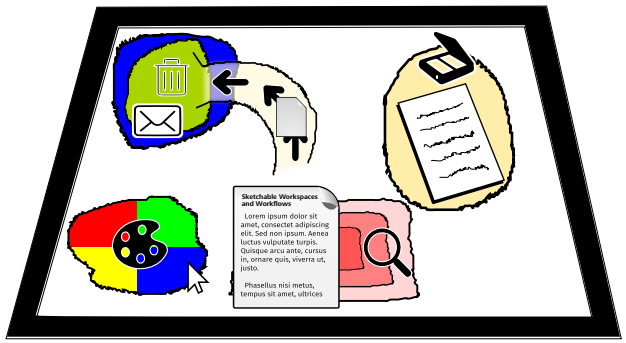

Figure 1: Sketchable Workspaces and Workflows allow users to customize and automate their digital workspace by drawing regions that affect objects placed on them in user-defined ways.

Workshop 'Rethinking Interaction' in conjunction with ACM CHI 2018, Copyright is held by the authors.

Abstract
We present "Sketchable Workspaces and Workflows" as a generic concept that allows end users to define workspaces and workflows on interactive surfaces by drawing regions that change how objects within these regions behave. To this end, we extend the *buffer* framework by Isenberg et al. in three ways: users can sketch interactive regions, these regions can effect actions that modify non-visual properties of objects, and those actions also apply to further objects such as windows, pointers, or physical objects. For example, a user might draw a rectangle on their computer screen and assign a *send to Tom* action to it. Any files which are dragged into this area are then mailed to Tom. This initial concept still requires iterating on design decisions and solving implementation challenges.

Author Keywords
none

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Motivation
In this position paper we present the concept of *Sketchable Workspaces and Workflows* based on the *Buffer Framework for Supporting Responsive Interaction in Information*

Visualization Interfaces (Isenberg, Miede, and Carpendale 2006). It allows users to define *action regions* on interactive surfaces by switching into a *sketching mode*. The action regions can be assigned one or more actions. These actions operate on any objects placed within the region - such as virtual and physical documents, windows, pointers, or tangibles. Actions include e.g., changing the size of a window, deleting or tagging a document, assigning a property to a pointer, or transferring information to a tangible. This set of basic actions allows end-users to visually define custom workspaces and workflows that automate common and special tasks.

Buffer Framework

In order to simplify development and improve performance of interactive information visualization interfaces on multi-touch tabletops, Isenberg et al. propose a buffer framework (Isenberg, Miede, and Carpendale 2006). In this framework, behavior of graphical objects is determined by their position on the screen. To this end, a *buffer* is provided for each property to be animated. In its most basic form such a buffer is a grayscale image with the same pixel resolution as the screen, similar to a *Z buffer* used in 3D rendering. Buffer types suggested by Isenberg et al. are e.g., *scale* or *translation*. For each graphical object on the screen, the buffer values directly under its center or anchor point determine the objects properties and behavior. For example, the value in the *scale* buffer determines the magnification of the graphical object on the screen (Figure 2). > Isenberg et al. discuss several improvements on this basic concept. For example, buffers may also be stored with lower spatial resolution whereby intermediate values are interpolated. Multiple buffers of the same type may also be combined, and buffers can be attached to tangible widgets on the tabletop. This allows for implementing e.g., a ‘magnifying glass’ by attaching a *scale* buffer to a tangible widget that can be dragged around the tabletop (Isenberg et al. 2007).

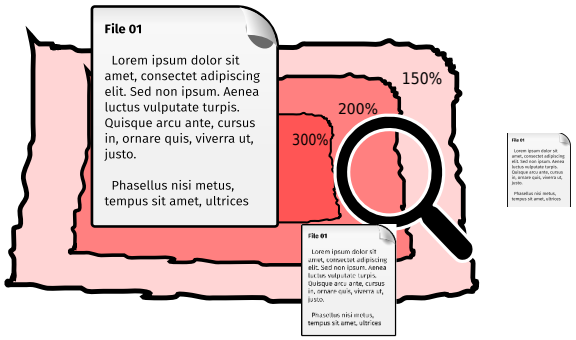

Figure 2: In the *buffer* framework by Isenberg et al., object properties (such as size) are affected by the values in (non-visible) buffers that cover the screen.

Extension 1: Sketching Regions

In the original framework the contents of all buffers are either defined by the application or by the position of a tool on the interactive surface. We make this system more flexible by allowing the user to switch the desktop into a *sketching* mode which offers a set of drawing tools. With these drawing tools, the user can paint values into the individual buffers, creating *active regions*. For example, the user might draw a path into the *translation* buffer that causes objects within the path to continually move around in a circle. A region with a gradient fill drawn into the *scale* buffer causes objects passing through it to temporarily change size (Figure 3).

Extension 2: Actions on Objects

In the original framework buffers only change the visual appearance of objects on the screen. We extend this concept

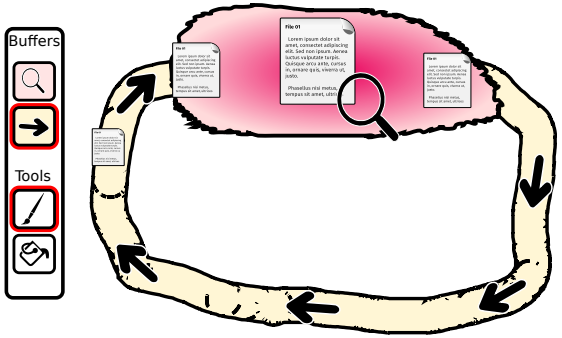

Figure 3: Using standard drawing tools for defining buffer contents allows for rapidly setting up workspaces and workflows. For example, a user can draw a carousel for their documents and a magnification area.

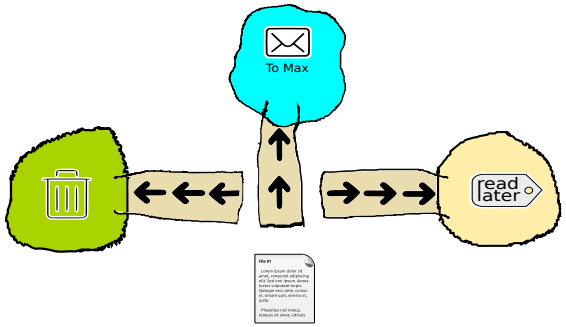

Figure 4: Action regions allow for applying transformations not only to graphical objects on the screen but also to their source files. For example, documents can be deleted, transmitted, or tagged by dragging their icons into one of the action regions.

by adding *action buffers* that operate on documents and other objects - not only their visual representations. When the visual representation of a document (e.g., a file icon or document preview) is placed within an active region that is assigned an action, the action is applied to the source document. Such actions might include *delete file*, *send by mail*, or *add a metadata tag* (Figure 4). Action buffers may also overlap, so that multiple actions can be applied simultaneously.

Extension 3: Further Objects

The action buffer concept can not only be applied to digital documents and their representations but also to other types of objects. This offers alternative ways to implement affordances found in common desktop environments (Figure 5). Windows moved into an active region could get maximized, closed, or magnified. By placing the mouse pointer on an active region within a color palette and clicking a button,

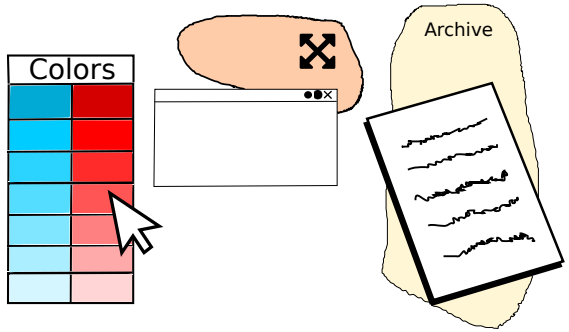

Figure 5: Actions can not only apply to digital documents but also to windows, pointers, or physical objects. For example, the user can change the color of the mouse cursor by clicking into a custom-drawn palette. Windows may be maximized by dragging them into an action region. Placing a paper document on a certain region scans it and archives a digital copy.

the pointer could gain an additional property *brush-color* and then be used for drawing on the screen. Physical documents - tracked on an interactive surface - could be tagged or scanned when placed within an active region. Tangible widgets could be placed into active regions that cause the clipboard contents to be copied onto the tangible widget.

Use Case

The interaction concept presented in this paper allows users to implement a multitude of workspaces and workflows by sketching active regions. For example (see Figure 1), a scientist reviewing a paper might scan in a printout of the paper which contains annotations by placing the sheets in a designated area of their interactive tabletop. The scanned document can then be dragged into the central area of the tabletop where a *scale* buffer magnifies the current page in order to ease reading and annotating. Placing the mouse pointer, a digital pen, or their finger on a field of the color palette selects the color used for annotating the paper. If the reviewer decides that the paper does not warrant publication, they gently push the virtual document onto the virtual conveyor belt they have drawn. The conveyor belt then moves the document into an action region that deletes the source file and sends an email to the author telling them that their submission was rejected.

Next Steps

In this paper we have presented a flexible concept for sketching custom workspaces and workflows. We are currently working on a proof of concept implementation of this concept. This initial implementation will be used to find sensible semantics for the drawing tools and the action regions. Further research is required for designing an appropriate user interface for sketching regions and adding custom actions. It is also necessary to investigate how well this approach does scale for more complex use cases - both regarding performance and usability. In the long term, we would like to integrate our concept into a standard desktop environment in order to investigate real-world uses and other interaction concepts (Wimmer and Hennecke 2010).

REFERENCES

Isenberg, Tobias, André Miede, and Sheelagh Carpendale. 2006. “A Buffer Framework for Supporting Responsive Interaction in Information Visualization Interfaces.” In *Creating, Connecting and Collaborating Through Computing, 2006. C5’06. the Fourth International Conference on*, 262–69. IEEE.

Isenberg, Tobias, Simon Nix, Martin Schwarz, André Miede, Stacey D. Scott, and Sheelagh Carpendale. 2007. “Mobile Spatial Tools for Fluid Interaction.” 2007-872-24. Canada: Department of Computer Science, University of Calgary. doi:1880/45785.

Wimmer, Raphael, and Fabian Hennecke. 2010. “Everything Is a Window: Utilizing the Window Manager for Multi-Touch Interaction.” In *Workshop ”Engineering Patterns for Multi-Touch Interfaces” in Conjunction with ACM EICS 2010*.