Augmented Interactions: A Framework for Adding Expressive Power to GUI Widgets

Jared Cechanowicz and Carl Gutwin

Department of Computer Science, University of Saskatchewan
110 Science Place, Saskatoon, Canada, S7N 5C9
jared.cechanowicz, carl.gutwin @usask.ca

Abstract. The basic elements of WIMP interfaces have proven to be robust components for building interactive systems, but these standard interactors also have limitations. On many occasions, researchers have introduced augmented GUI elements that are capable of more expressive interactions and that are better suited to user tasks. Although many of these novel designs have been effective, augmented interactors are still invented in an ad-hoc fashion, and there is no systematic way of thinking about or designing augmentations. As a result, there is little understanding of the principles underlying augmentation, the relationships between different designs, or the possibilities for creating new interactors. In this paper we present a framework that specifies elemental interactions with WIMP components and identifies the possible ways in which augmentations can occur. We show the explanatory and generative power of the framework by analysing existing and novel augmented interactions.

1 Introduction

The basic interactors of Windows-Icons-Menus-Pointer (WIMP) interfaces have proven over many years to be a robust and successful set of building blocks for developing interactive systems. As a result, the past 30 years have seen the standard desktop graphical user interface (GUI) change very little. While designs based on this model have been successful, a number of flaws have been identified (e.g., [2,4,16]). For example, desktop interfaces often require a large number of GUI widgets, with each widget mapped to a single system command. As a result, higher-level tasks like navigating and searching are not well supported, requiring that the user activate multiple controls, or manipulate a single control multiple times.

Numerous new controls, augmented controls, and novel interaction techniques have been developed that perform common desktop tasks better than standard WIMP interactors (e.g., [1,5,8,9,11,15,19,20,28]). As well, a number of augmented interactions that address WIMP limitations have been adopted as standard GUI idioms. For example, double-clicking is an augmentation of the basic click selection action, making use of a timeout period to increase the number of states that can be specified by the user; similarly, Shift-clicking is an augmentation that uses a key press to add a second mode to basic selection. Researchers have also introduced numerous augmentations, often based on an existing interaction or GUI widget. For example,

OrthoZoom [1] is an augmented scroll thumb that uses the unused horizontal dimension of mouse movement to control document zoom while the scroll thumb is activated. By enabling zooming as well as scrolling, higher-level navigation tasks are supported, and switching between separate scrolling and zooming widgets is no longer required. This augmentation, and others like it, have been shown to provide numerous benefits, including increases in user performance, better fit to high-level user tasks, and reduction in unnecessary screen clutter.

Although many different augmentations have been proposed, the design of these augmentations has most often been carried out in an ad-hoc fashion, and has usually focused on solving a particular interaction problem for a particular task. As a result, there is no standardized way of designing augmentations, and no way for designers to analyze or discuss the principles underlying an augmentation, relationships between different designs, or different design possibilities for creating new interactions.

In this paper we present a framework that is intended to provide this foundation for designers of augmented interactions. The framework identifies the core elements of an interaction in a WIMP interface, identifies the action primitives that can be used in an interaction, and specifies the types of augmentation that can be contemplated. The framework sets augmented interactions into a context of user tasks at the high level, and characteristics of input devices at the low level. The framework has both explanatory and generative power. We analyse and characterize several existing augmentations using the language and principles of the framework, and also use it to generate new augmented interactions that have not been seen before. The contributions of this work are: first, the idea that augmented interactions follow standard underlying principles; second, the framework that gathers these principles together in a form that is useful to designers; and third, a demonstration of the framework's power through several example designs. Our work shows that an understanding of the principles underlying augmented interaction can be a useful design tool, and can aid in the evolution of the GUI.

2. A Framework for Augmented Interaction

In order to simplify the process of designing augmented interactions for WIMP interfaces, we present a conceptual framework that is based on a high-level view of a user's interaction with a GUI. The framework has at its core the idea of an *interaction*, which we define as a combination of an *object* in the interface with one or more *actions*, each of which have a characteristic degree of freedom. Interactions are undertaken in service of a user *task*, and are supported by *input mechanisms* that provide the actual input data. In the following sections we describe each part of the framework in more detail, starting with the idea of an interaction.

2.1 Interaction: Object + Actions

A WIMP interaction can be defined as a user's manipulation of an on-screen entity. We formalize this with the concepts of the GUI *object* and the interface *action*; therefore, an interaction can be specified as one or more actions applied to an object.

WIMP Objects: Data and Controls

An object is an entity in a WIMP interface that has a visible representation. There are two basic object types in WIMP based GUIs: data objects and controls.

Data objects are the visual representations of the data of interest: icons in a file explorer, text and links in a web browser, or custom objects in a visual workspace.

Controls are graphical instruments that allow manipulation of data [2]. Since controls lie between the user's actions and the actual data, they are indirect instruments in a GUI. Traditional widgets such as buttons and sliders are the most common examples of controls; however, some types of data objects can also be given control capabilities (such as the links on a web page, which act both as data on the page, and as buttons that invoke navigation actions).

Actions in WIMP interfaces

Actions are the manipulations that are possible with a data object or control, and can be characterized by the degrees of freedom of the data that is being manipulated.

- *1D-Discrete*. The action is used to specify one of multiple states. For example, clicking on an icon in a file browser implies specifying which of two states the icon is in. 1D-D actions are often implemented with two-state devices such as mouse buttons, but devices with more than two states can also be employed [30].
- *1D-Continuous*. These actions allow specification of a single continuous value. For example, scrolling a document in one dimension is a 1D-continuous action. 1D-C actions can receive input from devices that are one-dimensional, but can also use a single dimension of a more powerful device (e.g., 1D scrolling using a 2D mouse).
- *2D-Continuous*. These actions allow simultaneous specification of two continuous values. An example action is 2D movement of a cursor; input is commonly received from any of several 2D pointing devices.
- *Higher-dimensional actions*. 3D and higher-degree actions are needed in some applications. However, they are not common in WIMP interfaces, and we do not consider these actions further, other to note that there are a number of high-degree-of-freedom input devices whose extra dimensions could be used in the augmentations described below.

Higher-level manipulations can be specified using these action primitives. For example, the common idiom of *dragging* can be characterized as an interaction made up of two actions: a 1D-D action (to select the object) plus a 2D-C action (to move it across the screen). Similarly, the idiom of 'Shift-clicking' can be characterized as a combination of two 1D-D actions: one for the shift, and one for the click.

2.2 Augmentation

An augmentation is a modification that is made to an action to increase expressive power; we identify several possible augmentations.

- *Adding states to a 1D-Discrete action*. A simple augmentation involves increasing the number of states that are possible for an interaction: for example, adding a state to an on-screen button changes it from a two-state widget to a three-state widget (e.g., pop-through buttons [30]).

- *Adding a 1D-Discrete action to an existing action*. Adding a discrete dimension to an existing action allows a multiplication of the expressiveness of the original – essentially adding modes to the interaction. Examples include common techniques such as Shift-clicking or Control-dragging, as well as research techniques such as Pressure Marks [26], which changes drag behaviour based on pressure level.
- *'Upgrading' a 1D-Discrete action to 1D-Continuous*. This allows the conversion of state-based manipulations to continuous manipulation. For example, a scroll button uses a 1D-D action; changing to a 1D-C action allows the scroll button to support variable-rate scrolling [2], given an appropriate 1D-C input source.
- *Adding a 1D-Continuous action to a 1D-Discrete action*. This augmentation can allow a continuous-value specification at the same time as a discrete selection. For example, Benko and colleagues developed techniques for continuous parameter control using finger position on a multitouch screen with bi-manual interactions [5].
- *Adding a secondary 1D-Continuous action*. Multiple dimensions can be controlled simultaneously with the addition of other 1D-C actions. For example, OrthoZoom [1] adds support for zooming (a secondary 1D-C action) to an existing 1D-C action (scrolling). Note that adding a second 1D-C action need not convert the interaction to a true 2D manipulation (e.g. horizontal and vertical scrolling); rather, it can remain a composite of two 1D manipulations [20] (as with OrthoZoom).
- *Adding a 1D-Continuous action to a 2D-Continuous action*. There are many ways that 2D movement can be augmented with an additional degree of freedom. For example, 1D-C pressure sensitivity is already used to control line thickness in many Tablet PC applications; pressure has also been used to control cursor size [26] and zoom during 2D pointer movement [23].
- *Adding a 2D-Continuous action to a 2D-Continuous action*. These augmentations add a second 2D capability to an interaction. Current examples generally involve the addition of a second 2D position controller – as seen in multi-touch displays which allow multiple fingers to simultaneously move, rotate, and scale objects.

As stated earlier, an interaction is made up of a GUI object and a set of actions. By adding to or modifying the actions related to an object, extra dimensions are added to the interaction which must be controlled by some input mechanism. In the following section we discuss input mechanisms as they relate to actions, and later discuss some additional rules for pairing input mechanisms and actions.

2.3 Input Mechanisms

Although a variety of input mechanisms can be used to control augmented actions, not every device is suited to every action, and choosing appropriate input is more complex than simply pairing devices and actions by the dimensions they control. The following paragraphs set out some of the issues in matching input device to actions.

Input Mechanism Properties

The properties of the input mechanism can guide the pairing of input mechanism and action, and here we highlight five properties that have been identified in previous research on input issues (e.g., [14,16,17]).

- *Physical Property Sensed*. Common properties sensed by input devices include position and force. Positional devices generally map best to positional tasks, and force has traditionally been used as a mapping for rate [14,17]. However, exceptions can be found: the mouse is used for rate control in Microsoft Windows, and pressure has been used for single-DoF positional control [7,22].
- *Absolute vs. Relative Mapping*. Absolute devices like sliders, and pressure sensors have a fixed 'zero' location, whereas a mouse and scroll wheel only sense relative movements. Relative devices are advantageous because they can be mapped to very large virtual spaces; however, they also require clutching. Absolute devices are best mapped to finite virtual spaces [17].
- *Continuous vs. Discrete Input*. Continuous devices like mice, foot pedals and pressure sensors map best to continuous tasks, but can also be quantized depending on the desired granularity [14]. Discrete devices provide the user with physical affordances, such as mechanical clicks and detents.
- *Reflexivity*. This is a property of absolute force-sensing devices like pressure sensors and isometric joysticks; these devices return to their zero position when released by the user. Reflexive devices avoid the 'nulling' problem [6] that can occur when an action is begun with the device not 'zeroed'.
- *Bi-directionality*. This is a property of relative devices like mice and scroll wheels; input can be specified as both positive and negative along a single axis. Some absolute devices have implemented bi-directionality by including a mode switch [25], or a second sensor [7].

Sources of Input

Depending on the properties of the action that must be supported, a number of input devices may be suitable for controlling the action. In situations where additional devices are impractical to add to the system, other input schemes can be employed. We have identified five ways that additional input capability can be obtained:

- *Overload the existing input capability with modes*. In this scheme, a discrete DoF facilitates a mode switch for another input. For example, holding down a modifier key (such as Shift or Control) could change the behavior of continuous actions (e.g., scrolling pages instead of lines with the scroll wheel) or discrete actions (e.g., open a link in a new tab instead of in the current window). FlowMenu [10] for example makes use of modes to increase the input capabilities of a stylus.
- *Use time as a DoF*. In this scheme time is used as a DoF. Time can be quantized and used to indicate discrete actions (e.g., 'hover help' activates after a time delay), as a continuous parameter for acceleration functions (e.g., scrolling accelerates the longer a scroll button is activated), and for mode switching (e.g., the difference between two clicks and a double-click). Time is commonly used in WIMP interfaces, and many gestural input systems use time as a DoF.
- *Use constraints.* In this scheme constraints are added to an interaction in order to create more complex behavior. For example, Kruger and colleagues [20] developed a constraint-based system called RotateNTranslate that allowed rotation to be calculated automatically from translation information. Similarly, Speed-Dependent Automatic Zooming calculates zoom level from the user's scrolling speed [15].

- *Leverage unused degrees of freedom*. In this scheme an unused DoF in the input device is used to control the augmented action. For example, Zliding [25] utilizes the unused pressure DoF to control zooming while sliding or scrolling with a stylus.
- *Add new degrees of freedom*. A final approach is to add new input capabilities to the input device to provide the needed degrees of freedom. Some upgrades take an existing device and transform it into a higher DoF device, as with the 6DoF VideoMouse [13]. Other upgrades to devices come in the form of independent input devices, as with pressure augmented mice [7] and the addition of the scroll wheel. Degrees of freedom can also be added to a system through independent modalities, including gaze [21], bimanual input [19] or continuous voice input [11].

At a minimum, an input mechanism must meet the dimension requirements of the interaction. However, higher-dimension input can be used for lower-dimension actions: for example, 1D-Continuous input could be quantized to provide a 1D-Discrete action, as frequently occurs when time is used as an input dimension.

2.4 User Task

Although specific tasks for augmented interactions will vary, there are several general reasons for wanting additional expressiveness during an interaction. We have identified four in particular:

- *Integrate interactions that make sense together or are part of a higher level task*. In some situations, additional tasks can be naturally combined with existing tasks. For example, scrolling and zooming are naturally combined into a navigation interaction [1,15], as are rotation and translation [20].
- '*Working with your hands full*.' In some cases it is important to provide alternate mechanisms for interaction when a primary mechanism is in use. For example, 'spring-loaded folders' allow users to open folders while dragging a file.
- *Integrate multiple single actions into a continuous control*. Frequent and repetitive single actions can often be reconsidered as continuous manipulations; for example, multiple presses of a 'Back' button could be converted into a multi-level 'Reach Back' button that goes back a variable distance. Ramos and colleagues' Pressure Widgets provide a similar interaction [24].
- *Allow richer input*. There are several situations where additional expressiveness could allow users to be more judicious in the execution of their tasks. Different types of richness include being able to express variable levels of selection (e.g., 'lightly selected,' 'strongly selected'), express variable levels of confidence in an action [8], or choose variable levels of preview. Many real-world examples exist – such as the way that the volume of a spoken command reflects its urgency: "open the door" versus "OPEN THE DOOR."

2.5 Augmentation Guidelines

The process of creating an augmented interaction, then, involves first identifying the action primitives that currently exist in the interaction of interest, augmenting the

actions, and choosing input mechanisms for controlling those actions. The framework makes it possible to consider augmentations as the application of simple changes to existing primitives, but the task that the interaction supports determines whether an augmentation is useful or needed.

In addition, although any number of augmented interactions are possible, not all augmentations would be effective or useful. When designing an augmented interaction, one can begin by describing the existing interaction in terms of the framework components: object, action(s), user task and input mechanism. By analyzing the interaction in terms of its parts, possible augmentations may reveal themselves. Comparing two similar augmented interactions in this manner can also reveal strengths and weaknesses in their respective designs, and potentially identify the more promising design. We have identified several issues that designers should consider when assessing the potential value of an augmented interaction:

- *Leverage natural mappings*. How a device is used can sometimes map naturally to the interaction itself. For instance, the rotation dimension of a Polhemus tracker maps easily to the rotation of an object, stylus hover maps to layers above the surface [9], and multi-state buttons can be used to indicate definiteness and confidence [8]. In addition, the direction of movement of the device and on-screen object / feedback should be compatible if possible [2].
- *Higher DoF is not always better*. Higher DoF actions can be useful in some situations, but troublesome in others. For instance, 2D drawing is accomplished with a mouse or stylus, but drawing a straight line (1D drawing) is difficult. As a result, programs include modes for locking an input dimension (holding Shift allows straight lines to be drawn). Even if the extra dimensions of the device are not used, a device that matches the degrees of freedom of the action is better suited to the task [2,16] (a 2D mouse performs better than a 6DoF device or two 1DoF devices in 2D pointing tasks).
- *Combine closely related interactions*. Some object parameters are naturally related (e.g. size and position, rotation and position) and suited to being combined in a single interaction [16]. OrthoZoom [1] and SDAZ [15] combine scrolling and zooming which are both important to navigating and reading documents.
- *Integrality vs. Separability*. When choosing an input mechanism, it may be unclear whether a higher-DoF device or two lower-DoF devices are more suitable. The principles of Integrality and Separability can assist when making this decision. Tightly coupled properties (e.g. size and position) are best controlled with a single high DoF input device, while separable properties (e.g. size and hue) are best controlled with two separate lower DoF devices [16].
- *Feedback*. All interactions should provide some form of feedback related to the state of the input device controlling the action. Some absolute devices, like foot pedals and sliders, already give some feedback to the user (both visually and through proprioception); however, visual feedback presented on or near the augmented GUI object is also important since the user's visual attention is on the object at the time of activation. Visual feedback is particularly important for pressure sensing devices [7,24]. Feedback through other modalities such as haptics [22] and pseudo-haptics [23] has proven useful in some cases for promoting user awareness of GUI objects.

4. Examples Using the Framework

In order to show the generality, expressive power, and generative capabilities of the framework, this section characterizes several augmented interactions using the concepts described above. We start with interactions that are commonly known in many GUIs, then characterize augmentations that have appeared in research literature, and finally present two interactions that are novel. These examples show that the framework is able to summarize a very wide range of existing augmentations, allowing them to be compared and discussed at an abstract level. In addition, the final examples show that the framework is valuable in the design of new augmentations.

4.1 Characterizing Common Augmented Interactions with the Framework

Here we look at two kinds of common augmented interactions: those using a keypress as a mode switch, and those using a time threshold to trigger enhanced behaviour.

Shift-clicking adds a mode to an existing selection action. The original action is a 1D-Discrete selection (often using the two states of the mouse button as input), and the augmentation uses a keyboard key, also a 1D-Discrete input device with two states: these two states imply two modes for the selection. (Users generally do not think of ordinary clicking as a mode, but *not* pressing the Shift key just means that the key is in its home state). Augmenting a selection action with Shift mirrors the key's original use as a single-mode augmentation of other keyboard actions (i.e., to provide capitals), but if the original action is carried out with an input device like the mouse, any key on the keyboard can be used as the augmentation input (as has been seen with variations such as Control-click or Alt-click; versions such as 'A'-click or 'B'-click are also possible, as long as these keys are not being used for text input).

Shift-dragging uses the same augmentation as shift-clicking, but with a different base action – dragging an object with the 2D-Continuous pointing device. The additional mode is often used to restrict the degrees of freedom of the base action from two dimensions to one – for example, limiting translation to use only horizontal or vertical movement. The augmentation can, of course, be used to increase capability rather than to restrict – for example, some pixel drawing programs use Shift-dragging to switch temporarily to the eraser tool.

Double-clicking is an augmentation of a single-click selection action (a 1D-Discrete action using the mouse button as an input device). The augmentation uses a second single click, separated with a time threshold (a 1D continuous input, discretized into two regions). This augmentation strategy provides a simple unary specification system, and can be extended: for example, triple-clicking is used in many applications (such as the Firefox browser), and higher numbers are possible.

Hover help augmentations also use time as the input mechanism. The base action is made up of two constraints – that the pointer (controlled by a 2D-Continuous input device) does not move, and that the pointer is located over a help-sensitive object. The time augmentation controls the appearance of the help balloon, which pops up if the pointer is held motionless for a certain time threshold (a 1D-Discrete action).

Spring-loaded folders are another 1D-Discrete augmentation using time as the input mechanism. Spring loading allows a user to open a folder in a file browser

without releasing the mouse button, and is an example of the 'working with your hands full' user task (see §2.4 above) that can be seen in standard file browsers in Macintosh OS/X and Windows Vista. The base action is a 2D-Continuous drag operation, with the added constraint that the pointer stops over a closed folder icon. The augmentation is a 1D-Discrete time threshold (as with hover help); once the threshold is reached, the folder under the pointer opens automatically. Using spring-loaded folders, however, reveals a usability issue when time is used as an extra DoF: setting an appropriate time threshold can be difficult, and folders can open too quickly (e.g., while the user is still reading the folder name or deciding whether this is the correct choice). The problem arises because time does not explicitly indicate user intention, and is often used for other purposes (such as reading the folder name). The framework's characterization of this interaction makes it clear that several other 1D-Discrete input mechanisms could be used instead of a time threshold; thus, the usability problem could be solved by using a mechanism that has a more explicit action that can be better interpreted as user intention (e.g., a keyboard key, or the secondary mouse button).

4.2 Characterizing Augmentations Proposed in Previous Research

Here we analyse several augmented interactions that have been proposed or evaluated in previous research literature. The framework allows characterization of a wide range of different designs, and also allows simple comparison of similar techniques.

Rate-controlled scroll buttons. Ordinary scroll buttons are widgets that control scrolling for a single axis of a document; clicking the button scrolls by one line, and holding the button scrolls at a fixed rate. The base action, therefore, is a simple 1D-Discrete action (selecting the widget with a two-state input device like a mouse button). The fixed scroll rate, however, is often not optimal for the user's task. To increase the expressiveness of the control, researchers have proposed augmenting scroll buttons to allow the user to control the scroll rate. This involves adding a 1D-Continuous action to specify the rate. There are several 1D-Continuous input mechanisms that can be used for this action: previous researchers have suggested a pressure sensor [3,24], but others are also possible, including pointer distance from the widget, or dwell time on the widget.

Combined zooming and scrolling. Researchers have invented techniques for controlling scrolling and zooming at the same time. In the OrthoZoom technique [1], one dimension of the 2D pointer controls document location (i.e., normal scrolling with the scroll thumb), and the other dimension (which is unused during scrolling) controls zoom level. In the Zliding technique [25], zooming with a pressure sensor augments ordinary scrolling. In these techniques, the base action of scrolling is a 1D-Continuous action on the scroll-thumb widget, and the augmentation adds a second 1D-Continuous action (the orthogonal dimension of the pointer or the pressure sensor) to control zoom. A third design that combines zooming and scrolling, but one that does not put zoom level under user control, is Speed-Dependent Automatic Zooming (SDAZ) [15]. In this technique, the zoom level is automatically calculated from the scrolling speed; although the augmentation still uses a 1D-Continuous action, there is no user control and the manipulation of the DoF happens entirely within the system.

Pop-through mouse buttons are a hardware augmentation of regular mouse buttons [30] that provide a second button click beneath the normal click mechanism. This converts a mouse button from a 1D-Discrete input device with two states, to one with three states (up, first click, second click). Pop-through buttons provide a novel input mechanism to match the addition of a 1D-Discrete action on an existing 1D-Discrete mouse-click-based action (such as the augmentation used in spring-loaded folders).

Bi-manual input. Several researchers have explored techniques that use two hands for input. For example, the non-dominant hand can control panning (a 2D-Continuous action), while the dominant hand performs detailed operations with the mouse [19]. Bimanual panning is an example of an augmentation that happens at the level of the workspace itself, and operates as a general capability that can occur along with any other operation (i.e., it is not specific to a particular interaction technique or widget).

Non-speech voice input has been suggested as a way to enrich standard desktop interactions. For example, Harada *et al* demonstrate 'Voicepen,' a drawing tool in which parameters such as line width or opacity are continuously controlled by the pitch or loudness of the user's voice [11]. The base action in this example (line drawing) is a 2D-Continuous action; the augmentations are separate 1D-Continuous inputs that control the different line parameters. The motivation for the augmentation is to increase the expressivity of standard drawing. This example is a good illustration of how the framework can assist designers with the comparison and evaluation of novel interactions. In the case of Voicepen, the framework's characterization suggests that the new input modality of non-speech vocalization could be compared to more traditional 1D-Continuous input mechanisms such as a slider or a mouse wheel.

4.3 Characterizing Novel Augmentations

To show the framework's value in helping designers explore new regions of design space, here we present two novel augmented interactions: one that allows users to control the size and detail of object previews, and one that allows users to specify an action with different degrees of confidence.

Variable-size previews. In this example, we show how the framework helped identify design opportunities in presenting richer previews (adding a 1D-Continuous action to control preview detail). Ordinary web links and icons represent underlying data, but do not fully describe it: for example, a file icon shows the type and name of a file, but not its contents; hyperlinks show even less, often indicating only that a link exists. To provide more information, some kinds of data objects provide previews of their content; however, these previews are usually provided at a single fixed size. We augmented the preview capabilities of ordinary object selection (a 1D-Discrete base action) to provide user control over preview size (Figure 1). This allows the user to select how much preview information is appropriate. We implemented this augmentation with a pressure-sensitive mouse; as the user presses harder on the button, additional detail is provided through a thumbnail image and a status-bar display. Pressure is a 1D-Continuous input mechanism, and matches the nature of the input action (i.e., requesting variable detail of preview). Other input mechanisms are also possible for this augmentation (e.g., a scroll wheel), but pressure maps well to an abstract idea like user interest [8], providing a natural mapping for the interaction.

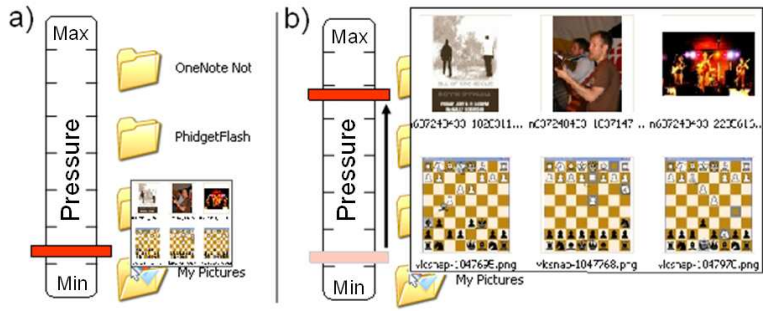

Fig. 1. a) Variable preview of a file folder. b) As pressure increases, thumbnail size increases.

Rich activations. Our second novel example involves the enrichment of user capabilities in a file explorer, through 1D-Discrete augmentations. Some actions in GUIs are possibly dangerous, such as opening system folders, or downloading items from the web that have been identified as potentially harmful. User preferences and system security settings often require that users confirm such activations through a confirmation dialog box, or may even require that users activate several menus to alter their preferences or security settings. This can result in user frustration, especially when a user's task is interrupted and when objects that the user knows are safe have been marked as potentially harmful. We augmented these activation actions with a 'degree of confidence' parameter that allows users to avoid unneeded confirmation dialogs. We used pressure for this new parameter's input, since pressure can be quantized into several different levels, and since (as described above) pressure maps well to degree of interest or confidence. With this augmentation, system folders can be opened without the dialog if the user applies pressure beyond a fixed threshold (Figure 2). Similarly, the user's confidence in activating web content can be communicated to the system through either a hard or a soft press of the mouse button.

This augmentation can also be applied to drag-and-drop operations (Figure 3). In this technique, the user can place content into a folder with a variable degree of confidence: for example, a user could drop a music file into a 'video only' folder by pressing harder before the drop action.

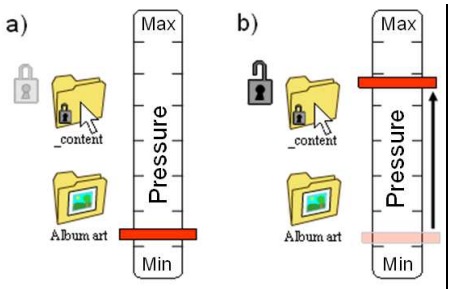

Fig. 2. a) A secure folder that requires enhanced activation. b) With enough pressure, the folder opens.

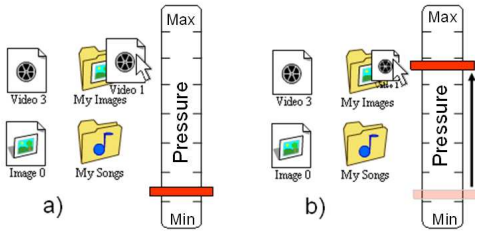

Fig. 3. a) The folder is set to only accept image files. b) With additional pressure, the icon shows that the defaults are overridden.

6. Discussion

Here we discuss the relationship of our framework to other formalisms and models of interface development, and comment on issues related to the design and use of augmented interactions more generally.

First, a number of other models exist for designing and developing interactions, including Direct Manipulation [29], Instrumental Interaction [2], and Reality-Based Interaction [18]. In addition, formalisms exist for specifying and notating interactions: for example, Buxton's three-state model [6] or the User Action Notation (UAN) [12]. Our augmented interactions framework is not meant to replace other design models; rather it is a tool for comparing and designing interactions that are developed in the context of other interaction models. Although we have presented this framework in the context of WIMP interfaces, the ideas can easily be applied to other interface-design paradigms. For example, an interface like CPN/Tools [3] does not include scroll bars, pull-down menus, or the notion of selection (instead, it includes a number of post-WIMP interactors like toolglasses, marking menus and floating palettes, as well as elements of direct manipulation). However, the augmented interactions framework could still be employed within this context: toolglasses could include multiple modes, or their size could be modifiable with an augmented interaction; floating palettes include buttons that could be augmented; and the direct manipulation actions in this interface could also be augmented using our approach.

The main contribution of our framework is that it looks explicitly at the issue of augmenting interaction, which extends what other formalisms are intended to do. For example, Buxton's three-state model can characterize and notate existing interactions, but does not set out what is possible for augmentation. Similarly, UAN is a notation for what does occur rather than a specification of what is possible; that is, a statement of action with a design rather than specification of the design space for a particular interactor. We note, however, that Buxton's model or UAN could be paired with our framework as notation.

Second, our experiences with augmented interactions suggest several questions regarding wider-scale deployment of these new techniques.

Will input hardware support the new designs? Additional degrees of freedom are gradually being added to input devices: scroll wheels are now standard, and commercial devices such as the IBM ScrollPoint mouse and the Xbox 360 controller support pressure input. Devices like isometric joysticks, pressure sensors, and multi-touch screens are widely available. As more powerful input devices become more readily available, more applications can make use of their capabilities.

Will new designs break existing interaction styles? One advantage of the framework is that it allows an augmentation to be broken into components so that designers can consider whether new actions can be supported with existing input devices. As shown in the examples, many augmentations can be designed such that the original interaction is preserved, and the augmentation can be used optionally (like a shortcut) by those who wish to do so.

Is the framework 'just for shortcuts'? Although shortcuts are a common modification, it is clear from the examples given above that the framework is able to characterize more than simply shortcuts. For example, adding rate control to a scroll button (a 1D-Continuous augmentation to a 1D-Discrete action) provides a degree of

control over scrolling that was not possible before; similarly, the ability to represent combined actions (such as scrolling and zooming) shows that the framework can help designers think about higher-level design ideas such as the integration of different kinds of behaviour in the same control.

7. Conclusions and Future Work

Augmentations to standard GUI interactions are now becoming common, both in research literature and commercial products; however, most augmentations are designed in an ad-hoc fashion. We presented a framework for understanding and designing augmented GUI interactions that can aid in comparing, evaluating, and building GUI augmentations. The framework is able to categorize and describe a wide range of previously developed augmented interactions. We also presented augmentations that are novel, showing the power of the framework to help in the exploration of design space and in the identification of new design opportunities.

Our work in the area of augmented interactions will continue in three ways. First, further development and refinement of the framework will add detail to the basic dimensions described here. Second, quantitative evaluations of some of our designs will be carried out to measure the benefit of various augmentations and to test the comparative power of the framework. Third, we will explore possibilities in toolkit support for augmented interactions, so that designers can quickly and easily include augmented interactions in new applications, and also retrofit existing systems.

References

1. Appert, C. and Fekete, J. OrthoZoom scroller: 1D multi-scale navigation. In: ACM Human Factors in Computing Systems, pp. 21--30 (2006).
2. Beaudouin-Lafon, M. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In: ACM Human Factors in Computing Systems, pp. 446--453 (2000).
3. Beaudouin-Lafon, M., and Lassen, H., The Architecture and Implementation of CPN2000, a Post-WIMP Graphical Application. In: ACM Symposium on User Interface Software and Technology, pp. 181--190 (2000).
4. Beaudouin-Lafon, M. Designing interaction, not interfaces. In: ACM Advanced Visual Interfaces, pp. 15--22 (2004).
5. Benko, H., Wilson, A. D., and Baudisch, P. Precise selection techniques for multi-touch screens. In: ACM Human Factors in Computing Systems, pp. 1263--1272 (2006).
6. Buxton, W. Lexical and pragmatic considerations of input structures. ACM Comput. Graph. 17, 1, 31--37 (1983).
7. Cechanowicz, J., Irani, P., and Subramanian, S. Augmenting the mouse with pressure sensitive input. In: ACM Human Factors in Computing Systems, pp. 1385--1394 (2007).
8. Forlines, C., Shen, C., and Buxton, B. Glimpse: a novel input model for multi-level devices. In: ACM Human Factors in Computing Systems, 1375--1378 (2005).
9. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., and Balakrishnan, R. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In: ACM Human Factors in Computing Systems, pp. 861--870 (2006).

10. Guimbretiére, F. and Winograd, T. FlowMenu: combining command, text, and data entry. In: ACM User Interface Software and Technology, pp. 213--216 (2000).
11. Harada, S., Saponas, T. S., and Landay, J. A. Voicepen: augmenting pen input with simultaneous non-linguistic vocalization. In: International Conference on Multimodal Interfaces, pp. 178--185 (2007).
12. Hartson, H., Siochi, A., and Hix, D., The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs. ACM ToIS, 8 (3), 181-203 (1990).
13. Hinckley, K., Sinclair, M., Hanson, E., Szeliski, R., and Conway, M. The VideoMouse: a camera-based multi-degree-of-freedom input device. In: ACM User Interface Software and Technology, pp. 103--112 (1999).
14. Hinckley, K., Input Technologies and Techniques. In: A. Sears and J. Jacko (eds.) The Human-Computer Interaction Handbook, pp. 3-11, CRC Press (2002).
15. Igarashi, T. and Hinckley, K. Speed-dependent automatic zooming for browsing large documents. In: ACM User Interface Software and Technology, pp. 139--148 (2000).
16. Jacob, R. J., Sibert, L. E., McFarlane, D. C., and Mullen, M. P. Integrality and separability of input devices. ACM ToCHI, 1, 1, 3—26 (1994).
17. Jacob, R., The future of input devices. ACM Comput. Surv. 28, 4, 138--159 (1996).
18. Jacob, R., Girouard, A., Hirshfield, L., Horn, M., Shaer, O., Solovey, E., and Zigelbaum, J. Reality-based interaction: a framework for post-WIMP interfaces, In: ACM Human Factors in Computing Systems, pp. 201--210 (2008).
19. Kabbash, P., Buxton, W., and Sellen, A. Two-handed input in a compound task. In: ACM Human Factors in Computing Systems, pp. 230--238 (1994).
20. Kruger, R., Carpendale, S., Scott, S., and Tang, A. Fluid integration of rotation and translation. In: ACM Human Factors in Computing Systems, pp. 601--610 (2005).
21. Lucas, J., Kim, J., and Bowman, D. Resizing beyond widgets: object resizing techniques for immersive virtual environments. In: ACM Human Factors in Computing Systems, pp. 1601--1604 (2005).
22. Miller, T. and Zeleznik, R. The design of 3D haptic widgets. In: ACM Interactive 3D Graphics, pp. 97--102 (1999).
23. Mandryk, R., Rodgers, M., and Inkpen, K. Sticky widgets: pseudo-haptic widget enhancements for multi-monitor displays. In: ACM Human Factors in Computing Systems, pp. 1621--1624 (2005).
24. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure widgets. In: ACM Human Factors in Computing Systems, pp. 487--494 (2004).
25. Ramos, G. and Balakrishnan, R. Zliding: fluid zooming and sliding for high precision parameter manipulation. In: ACM User Interface Software and Technology, pp. 143--152 (2005).
26. Ramos, G. A. and Balakrishnan, R. Pressure marks. In: ACM Human Factors in Computing Systems, pp. 1375--1384 (2007).
27. Rekimoto, J. and Schwesig, C. PreSenseII: bi-directional touch and pressure sensing interactions with tactile feedback. In: ACM Human Factors in Computing Systems, pp. 1253--1258 (2006).
28. Ren, X., Ying, J., Zhao, S., Li, Y. The Adaptive Hybrid Cursor: A Pressure-based Target Selection Technique for Pen-based Interfaces. In: IFIP Conference on Human-Computer Interaction, pp. 310--323 (2007).
29. Shneiderman, B. Direct Manipulation: a Step Beyond Programming Languages. IEEE Computer, 16, 8, 57--69 (1983).
30. Zeleznik, R., Miller, T., and Forsberg, A. Pop-through mouse button interactions. In: ACM User Interface Software and Technology, pp. 195--196 (2001).