Collection Objects: Enabling Fluid Formation and Manipulation of Aggregate Selections

Haijun Xia, Bruno Araujo, Daniel Wigdor
University of Toronto
{haijunxia | brar | daniel}@dgp.toronto.edu

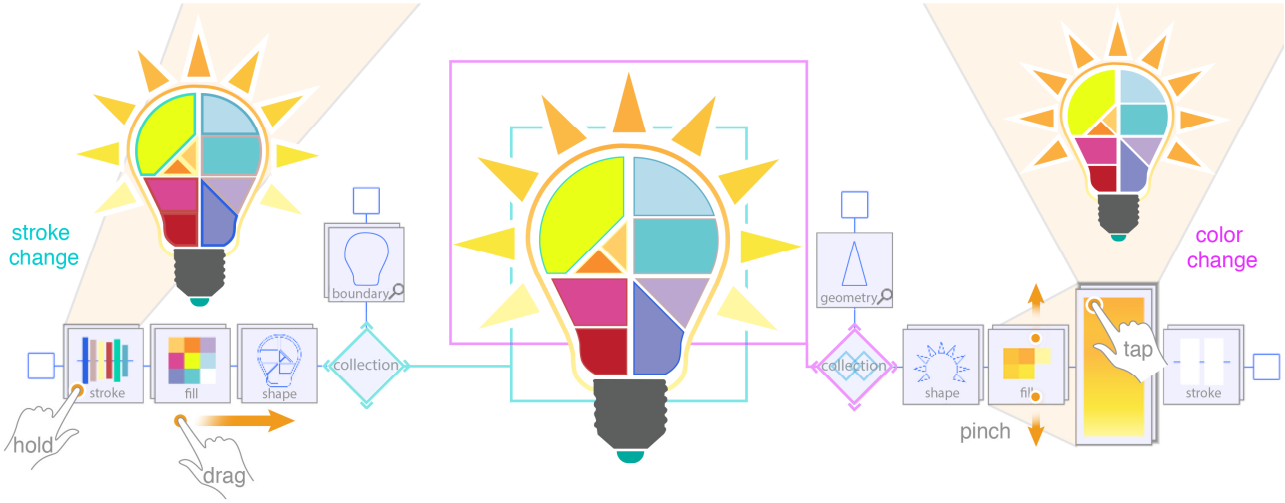

Figure 1. A drawing of a lightbulb, comprised of a number of drawing objects, is manipulated using Collection Objects. Left: the thickness of all strokes is adjusted simultaneously. Right: The color of the "rays" is harmonized with a single gesture.

ABSTRACT

Despite the long development of Graphical User Interfaces, working with multiple graphical objects remains a challenge, due to the difficulties of forming complex selections, ambiguities of operations, and tediousness of repetitively unselect-reselect or ungroup-regroup objects. Instead of tackling them as individual problems, we attribute it to the lack of system support to the general selection-action cycles. We propose Collection Objects to not only support a single fast selection-action cycle but also allow multiple cycles to be chained together into a fluid workflow. Collection Objects unifies selection, grouping, and manipulation of aggregate selections into a single object, with which selection can be composed with various techniques, modified for later actions, grouped with objects inside still directly accessible, and quasi-moded for less context switching. We implemented Collection Object in the context of a vector drawing application with simultaneous pen and touch input. Results of an expert evaluation show that Collection Objects holds considerable promises for fluid interaction with multiple objects.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CHI 2017, May 06-11, 2017, Denver, CO, USA
© 2017 ACM. ISBN 978-1-4503-4655-9/17/05…$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025554

INTRODUCTION

Direct-touch and pen input offer several advantages over traditional indirect input, including mobility and the replacement of dedicated control surfaces with larger screens [17]. Due to these advantages, direct touch has achieved near ubiquity in higher-end mobile devices. As these devices have become popular, UIs have been adapted for direct touch. The use of direct physical manipulation, enabling control through translation, rotation, and scaling with simple gestures, rather than by selecting and manipulating offset controls, is now nearly universal in such devices. One particular advantage of gesture-based systems is the ability to *phrase* object selection, operations, and additional parameters into a single gestural stroke, resulting in fewer mental steps [10]. Although desirable, it is brittle: mental flow breaks whenever any step requires excessive cognitive effort. Surprisingly, a key operation has resisted being made fluid: multiple object selection. Such selections have been called-out as a particularly tedious subtask in direct touch UIs [14, 27].

Problems with selecting and manipulating multiple objects have been identified since the dawn of the direct manipulation paradigm [13, 37]; it is perhaps, therefore, unsurprising that improving selection has been a focus of considerable effort in research. Less studied, however, has been the user's experience immediately before and after engaging a particular selection technique, and how well the technique flows as part of a phrase of related gestures. It is the thesis of the present work that no matter how sophisticated a technique may be, users suffer if it requires a significant shift in focus from their primary activity.

In its most common form, selection is a transient state, and each newly made selection replaces the previous one. While this reduces the use of modes, it can also cause significant frustration. For example, a user may lose the work performed theretofore that they composed a large aggregate selection due to an error late in the process [30]. Commonly, selections may be saved as groups for later reuse, but rigid group structures often force users to constantly select and reselect, as well as ungroup and regroup while switching attention among objects [32]. There is an opportunity, therefore, to improve workflows by changing selection to be more object-like, while simultaneously providing more flexibility.

In addition to the difficulties of selecting multiple objects, the WIMP UI also falls short when visualizing and interacting with the selected objects. In noun and verb pairings enabled by toolbars, which act as the core of WIMP UIs, when two objects have different values for a field, the field is typically left blank. This creates ambiguity within the interface [19]. Moreover, select+toolbar also fails to provide support for simple workflows when working with multiple objects. If a user wants to share the color of an object to a set of selected objects, for example, she will have to 1) deselect the set of objects, 2) select the object which has the color, 3) look up and copy its color value, 4) reselect the objects, and 5) finally apply the change.

Our goal is to facilitate interaction with multiple objects on touchscreen devices, while maintaining the direct physical manipulation paradigm. We approach this from two directions. First, we seek to replace transient selection and rigid groups with an embodiment that can be manipulated directly, resulting in lower penalties for erroneous selections and easier reuse. Second, we seek to explore how various selection, grouping, and manipulation techniques can be encapsulated within such an embodiment to enable a continuous workflow with less mode and context switching.

We developed Collection Objects (COs). COs embodies selections and groups as objects, which can themselves be directly manipulated using gestures. As such, selection is no longer a transient state of objects but a collection of objects that can be filtered, overlapped, and collectively manipulated. Further, a group is no longer a rigid and closed structure, but rather a set of objects that can be freely added, removed, navigated, and accessed directly.

Built upon Attribute Objects [44], COs consists of two sets of attributes: 1) attributes acting as expressive search filters to enable rapid expansion and contraction of the selection scope and 2) attributes enabling for meaningful visualization and the advanced manipulation of multiple objects within the collection. Complementary to the ***micro*** interaction with attributes of an object [44], our work, explores ***macro*** manipulation of multiple objects. We demonstrate how the various properties of COs can be coherently encapsulated to achieve continuous and fluid manipulation of multiple objects in a vector drawing application. We also present the results of an expert assessment, wherein professional illustrators used our drawing tool to create vector arts.

RELATED WORK

Our research draws from prior work on selection techniques for gesture-based interfaces, alternative selection mechanisms and grouping, the manipulation of multiple objects in GUIs, and the objectification of UI elements.

Selection Techniques in Touch and Pen based Interfaces

Selection is a key research problem in HCI. Significant work has investigated the efficiency of different input devices for pointing tasks [5, 12, 25]. Although relevant, the goal of the present work is to reduce the number of operations required to make and apply operations to selected groups of objects. This is intended to increase the efficiency of user input, which will save time and reduce fatigue and tedium. This is especially true for touch, which suffers from the "gorilla arm" (fatigue) [8] and "fat finger" (small target selection) problems [34].

The present work falls into the area of interaction techniques for direct-touch systems. A number of projects have examined selection for such devices. Accot and Zhai found that the performance of stylus crossing to select multiple targets depends on the layout and size of the targets [1]; similar results for touch input have also been found by Luo and Vogel [24]. Leitner and Haller further combined stylus crossing with area cursors to enable complex selections[22]. Strothoff et al. studied pin and touch gestures for efficiently grouping objects [36]. Tse et al. also explored gestural selection enabled by wrapping their hands around the objects to be selected, although this required more detail about user touch areas than most capacitive touch sensors report to software [39, 41].

Selection using pen-based UIs has also been widely explored [14, 22, 27]. Scriboli employs delimiters to phrase selection and action into one continuous pen stroke [14]. Recently, Hinckley et al. [15] advocated for a division of labor between pen and touch based on the strengths of each input modality: *pen writes and touch manipulates*, which built on findings from Brandl et al. [9]. Our work builds on these, with the selection and manipulation of many objects being achieved with direct touch manipulation gestures.

Recently, research has sought to ease the difficulties of selection by providing the user with system-interpreted selections. Suggero [23] and cLuster [29] employed Gestalt principles and various pre-trained selection perspectives to suggest possible selections for users based on their initial selection. Sloppy selection [21] and Lazy selection [45] refined users' selection based on user input and the underlying pattern of the screen content.

Our work is complementary to all the above techniques; we seek to provide a single selection mechanism, which can coherently encapsulate various selection techniques and allow users to fluidly compose, modify, and reuse a selection by manipulating a set of filter attributes.

We look beyond selection techniques, seeking to enable fluid selection-action flow which improves the efficiency and experience by reducing the number of tedious and unnecessary selection tasks.

Alternative Selection and Grouping Mechanisms

In a typical GUI dialog, a selection is created to indicate to the computer to which objects a command will apply [30]. Usually, when a new selection is made, the previously selected objects are deselected. It is common, however, that particular parts of content require frequent editing, resulting in the tedious repetition of selection tasks. To ease this problem, some commercial design applications, such as Adobe Illustrator, allow users to save a selection before it is replaced by a newly made one. However, this requires the user to foresee its usage and navigate through menus. Anecdotally, this has been referred to as a "hidden gem" of Adobe Illustrator [31].

Grouping is an alternative to reusing a previous selection. Typically, a user can group currently selected objects to transform the selection of multiple objects into a persistent aggregate object. As such, the selection of any object inside the group always selects the group as a whole. While this leads to easy reselection of the same set of objects, the rigid structure prevents fluid selection across groups which forces users to constantly group and ungroup objects [32].

Additionally, the hierarchical structure of groups prohibits the simultaneous existence of overlapping groups. To address this, Saund et al. proposed a flat lattice grouping structure to allow users to create overlapping groups [32]. A sequence of groups associated with one object can be cycled through by subsequent mouse clicks according to the most-recently-used order. Similarly, Rock & Rails [42] proposed creating offset proxy objects, where manipulations performed on the proxies were applied to both the proxies and their linked objects. Further, objects were able to belong to multiple groups simultaneously, by linking to multiple proxies. Our work combines hierarchical structure, which allows for meaningful organization of objects, and overlapping selection, which penetrates the hierarchical structure and allows for high flexibly of selection. We further reduce the rigidity of hierarchical structure by enabling easy and fast navigation of levels of hierarchy with direct bimanual gestural input.

Manipulation of Multiple Objects in GUIs

In a traditional toolbar UI, the values of a selection's attributes are displayed when one object is selected. However, when multiple objects are selected, only those shared values are displayed, and the visual representation for the remaining ones is ambiguously missing [19]. Prior work has sought to solve many of the problems of manipulating the attribute values of multiple objects [13, 16, 19, 33]. Hoarau et al. display all the attributes of the selected objects in a side panel to reveal the implicit structure of a selection [16]. Interactions with such values enables limited manipulation of the subsets of objects represented by the values. Kwon et al. provide a UI with both numeric and direct manipulation handles on surrogate objects [19]. Our work provides a novel UI solution to tackle this problem that is faster and a more direct means to accomplish such tasks, enable new ones, and avoid tedious interactions.

In a command-object system, one's experience manipulating multiple objects can be further enhanced with advanced commands. GACA employs a group-aware arrangement tool to reduce the number of alignment operations [45]. We support the typical workflow of interacting with multiple objects, such as sharing various attributes to a set of objects and editing individual objects without paying the penalty of switching selections, while preserving the space wherein advanced operations of each attribute can be embedded.

When interacting with multiple objects, users strategically plan their operations with respect to the given system for optimal performance [26]. As an example, marking menu favors object-oriented strategy: it is more efficient for users to issue multiple distinct commands on one object. The opposite, floating menus are more efficient when a user repeatedly issues the same command, favoring command-oriented strategy. While a common UI design approach is to favor one or another and let users adapt to it [30], users have to constantly shift between strategies based on the nature of tasks, their cognitive contexts, and individual preferences [26]. The mismatch between the user strategy and the interaction technique breaks user's continuous mental flow and results in lower task performance. Our approach supports both strategies: it enables users to change individual attributes of all selected objects, while it still provides the flexibility to change every attribute of individual objects within a selection without context switching.

Objectification of UI Elements

The present work seeks to enable complex editing while maintaining the direct physical manipulation paradigm. To do so, we introduce a new UI element which embodies the selection and grouping, and can be directly manipulated. This approach directly builds on several projects attempting to objectify UI elements.

Researchers have made several attempts to objectify interaction into GUI controls with the intention of leveraging the human experience with the physical world to interact with the digital one [35]. A traditional approach is to objectify tools or commands using graphical UI widgets such as buttons or menus. By providing an object-like representation to tools, they become objects that can be directly manipulated by users [2, 4, 7, 28]. While this led to interfaces with familiar elements borrowed from the physical world, one drawback is that such interactions are limited by their reliance on metaphors: the physical world provides no mechanism to interact with abstract content, such as the brightness or opacity of a photo. As an example, in most present systems, attributes are represented using numbers or text, and these attributes confine the interactions to basic control widgets, such as buttons and slide bars [3].

To extend the usage of direct physical manipulation in GUIs, Xia et al. [44] proposed *Attribute Objects* as an objectification of abstract attributes. Using a card-like representation, Attribute Objects can be moved, copied, linked, interpolated, and so on, using direct manipulation gestures. The prototype application, Object-Oriented

Drawing (OOD), demonstrated how complex drawing applications could be built without the need for complex, form-based UIs. Although OOD enables micro-manipulation of every attribute, a clear omission is the ability to macro-manage multiple objects. Our work builds on the concept of objectifying abstract content: each of our Collection Objects is the embodiment of the selection state and the group structure. We demonstrate how such an embodiment provides a set of powerful interactions with multiple objects.

COLLECTION OBJECTS

Despite the simplicity of the two step selection-action model of a GUI dialog, users encounter a number of difficulties when interacting with multiple objects. A selection, while being formed, may get lost due to its transient nature; it may also be incomplete or hard to complete as objects fall out of the viewport or hide under other objects. Objects can be grouped to maintain the structure or to reuse the selection, but the rigid group structure prohibits the selection of individual objects within it, resulting in repetitive ungroup-select-regroup procedures. Navigation and the manipulation of group structure is poorly supported, often causing a user to break her actions and seek help from other widgets (e.g., a layer panel). While in multi-selection, quickly copying or adjusting the style of an object within the selection requires a user to switch between multi-selection and single selection; thus, what is only one mental step for the user cannot be achieved without several selection-action cycles.

Collection Objects attempt to address all the above problems by providing a single coherent UI with the following properties.

It objectifies selection. It appears as an independent object on the canvas, which is persistent: it must be deleted or modified to end the selection. As such, existing selections will not get lost unless the user explicitly deletes them, resulting in a lower penalty for erroneous selection and easier reuse of a previous selection without creating rigid structure.

It can be composed and modified. Collection Objects coherently encapsulates various selection techniques as well as afford new ones using a set Attribute Object [44]. Attributes such as geometry and color can be used as selection filters to search for, and select, objects with the same style across the entire canvas. Attribute Objects of Collection Objects may be added, replaced, manipulated, or removed using simple direct manipulation gestures, enabling a rich set of selection techniques and rapid expansion and contraction of the selection scope.

It objectifies group structure. Objects in a collection can be grouped only for maintaining the structure. Because of objectification, the scope of a group can be trivially managed using the same techniques of composing selection, which eliminates indirect management with other widgets and context switching. Group structure is hierarchical to enable meaningful content organization. Yet, it can be quickly navigated using bimanual gestures to edit its subparts.

It integrates overlapping selection**,** which allows for simultaneous existence of meaningful and flexible selections, ***with open group structure*****,** which further grants selection of objects across groups, while preserving the hierarchical structure.

It supports rich manipulation of multiple attributes. Tuned for the typical workflow with multiple objects, Collection Objects support the quick sharing of attributes to a collection and the manipulation of individual objects by quasi-moding the collection. By doing so, one can fluidly conduct a sequence of actions without switching between single selection and multi-selection.

It requires no context switching. By encapsulating the above functionalities, sequence selections and actions take place within one UI element**.**

Creating and Deleting Collection Objects

A Collection Object (Figure 2) is instantiated when the user wants to work with multiple objects simultaneously. This can be done by holding a finger on the canvas. Once initiated, *tapping* drawing objects while holding a Collection Object adds them to that CO. Alternatively, a user can expand from a single selection using the miniature CO (Figure 3) on every single object. The selection can be deleted by dragging the Collection Object off the canvas.

Identifying and Representing Collection Objects

Each Collection Object is represented by a diamond (Figure 2). Since several Collection Objects can co-exist on the canvas, it is important to differentiate each CO, as well as to be able to easily find a particular one. To do so, the border of the CO is color-coded. When manipulating one object, its existing Collection Objects, as well as the link between them, are highlighted using the corresponding color.

The diamond provides an anchor, where each of its four corners represent a different attachment. The left corner links to the drawing objects contained within the collection, visually linking the CO and its constituent objects. The top corner of the diamond attaches to the set of Attribute Objects which act as the attributes of the CO and define a filter for the selection. Additionally, another set of Attribute Object cards is located on the right corner of the diamond. These represent the *attributes of the objects* contained within the selection, similar to the toolbar in a traditional UI. Adding to, removing from, or changing Attribute Object cards in this set will result in changes to the selected objects. Finally, the bottom corner can be connected with another CO to spawn a child collection using Boolean operations.

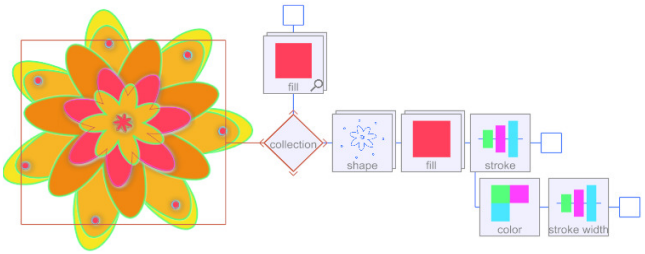

Figure 2. A drawing of a flower. Its red pedals have been collected in the Collection Object shown to the right. Two sets of Attribute Object cards are shown: Those of the Collection Object (top), and those of the selected objects (right).

SELECTION WITH COLLECTION OBJECTS

Replacing the transient selection state with a persistent embodiment, Collection Objects, enables safe iterative selection without the fear of accidentally losing it with an errant click. Just as important, it also allows for fluid cycles of selection-action units by reusing and modifying existing Collection Objects. In addition to simply holding the CO and tapping objects to be added into the collection, Collection Objects unifies various selection techniques, such as lasso/crossing selection, selection by attribute, suggestive selection, and fuzzy selection. Further, because selections are persistent, each may be used to iterate on the current selection.

Attributes of the Collection Object

Attributes of the Collection Object located on the top corner of the diamond act as filters to define the selection (Figure 2) A user can add filter attributes into the collection by dragging attribute cards into the filter list. Attributes in this set are applied to the selection using an "and" relationship, similar to the chain effect in Side Views [38]. A user can expand or shrink the scope of the selection by adding or removing Attribute Object cards to this set (Figure 3). We now describe the unique set of filter attributes of our Collection Objects and their effects on selection tasks.

Selection Boundary Attribute

Lasso selection and cross-to-select with pen input are common selection techniques in gesture-based interfaces [1, 14, 24]. To support this, each CO contains a boundary attribute, which enables both selection techniques. To create a boundary attribute, the user first holds the CO to put the system into a quasi-mode, where the paths drawn by the pen become selection boundaries rather than normal drawing paths. In this quasi-mode, as soon as the pen touches the screen, it creates a boundary attribute which is inserted into the CO. The shape of the boundary determines which drawing objects are selected: a closed (or nearly closed) boundary selects objects inside it (lasso selection), while an open boundary selects objects it crosses (crossing selection). A user can further edit the boundary by holding the attribute and modifying the paths, similar to [22].

Alternatively, the shape attribute of existing drawing objects may be used as a selection lasso to select all the drawing objects within its bounds. This reduces the need to create a selection boundary, which can be difficult in a crowded scene. Figure 3 shows an example utilization: to select all of the strokes comprising the alpaca's maw, the user simply opens the stack of Attribute Object cards for the outline of the maw. Similar to the original OOD, the user can make a copy of its shape attribute and drag it into the Collection Object defining a boundary. All objects contained within the boundary are selected immediately.

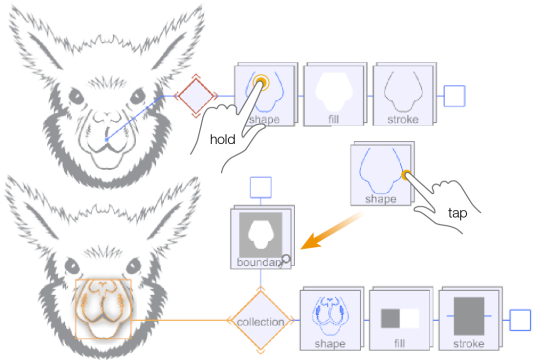

Figure 3. The user copies the "shape" attribute of the maw and drags it into a CO, which uses the shape as a kind of lasso, immediately selecting all drawing objects within it.

Style Attribute

Other attributes of existing objects may also be copied and inserted into the Collection Object to enable selection by attributes. For example, if a user wishes to select all objects with a particular fill color(s), she may copy the fill attribute of an existing object and insert it into the CO. Immediately, all drawing objects with that same fill color are selected (Figure 4 a,b). Similarly, in a night scene where the stars share a shape but not a color, a user can select all the stars by inserting a geometry attribute (i.e., a sub-attribute of a shape, regardless of size, orientation, or position) into the Collection Object. Instead of making a selection and editing the selected objects, a user can dynamically change the selection scope by manipulating the values of filters. The selection feedback of different attribute value reveals how attributes are used across the scene. In a floor plan example, a user can adjust a stroke width filter attribute to understand the load distribution on the wall.

Fuzziness Meta Attribute

With vector graphics, it is common that different drawing objects have attributes that are similar but not identical, such as color or stroke width. When an Attribute Object is inserted into the Collection Object, a fuzziness sub-attribute is automatically added to it. The fuzziness value indicates the tolerance associated to a filter and can be edited to handle similar but not identical attributes (Figure 4). For example, in a heat map, searching by a particular fill color with a certain tolerance allows the user to select objects whose fill is within the chosen range across the canvas. She can also adjust the tolerance to dynamically change the selection. This can be useful, if, she wishes to quickly locate similar objects and change them collectively.

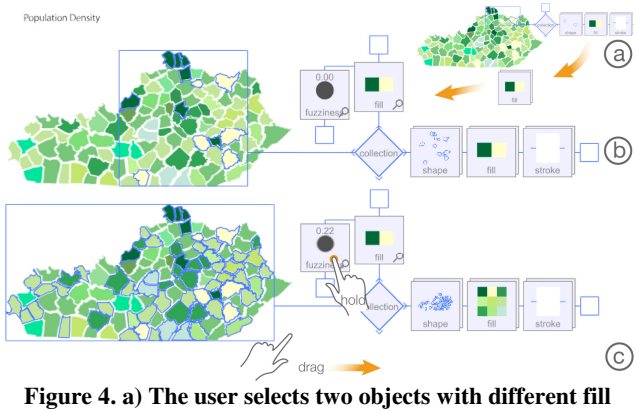

Figure 4. a) The user selects two objects with different fill colors b) and uses the fill to select objects with either color; c) she can then adjust the fuzziness sub-attribute to select objects with a similar but not identical color.

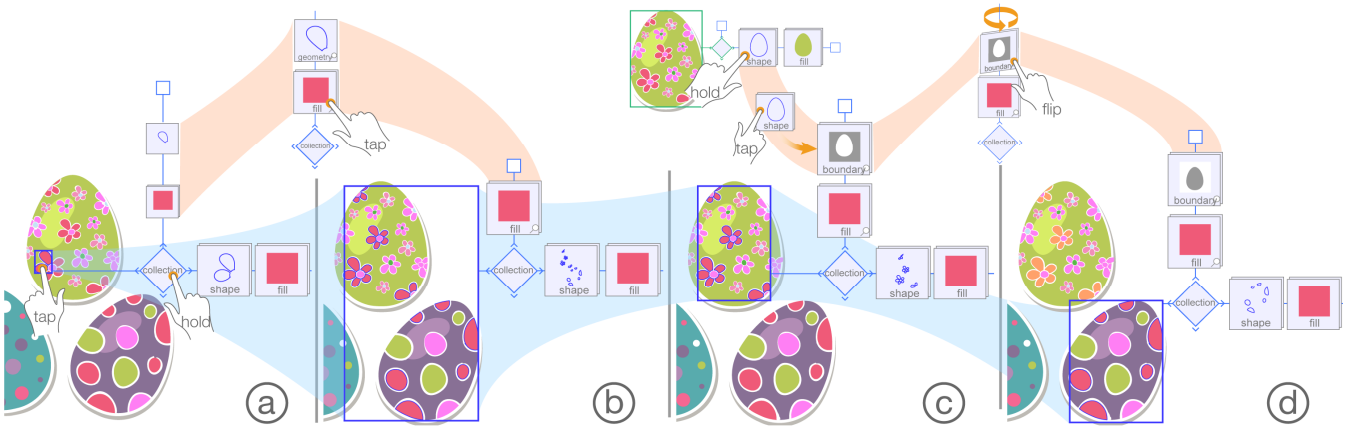

Figure 5. Selection sequence on one Collection Object. a) the user holds a CO and taps pedals to select them; the system automatically suggests possible filters; b) tapping the fill expands the selection to all red objects; c) she can copy the shape of the top egg and drag it into the CO to limit the selection; d) after changing the color, flipping the boundary card selects the red pedals in another egg.

Emergent Filter Attribute

In some cases, a user may start to select objects individually, without care for their attributes. This can be accomplished by holding the CO and either tapping the object to be added or lassoing a selection boundary. As the user adds objects, the tool will examine them for common attributes and then recommend the commonalities as possible criteria. These emergent selection filters appear briefly on the filter attribute list with a lower opacity. A single tap on the potential attribute adds it persistently to the collection and will expand the selection scope to include all objects with the same attribute value (Figure 5a,b). This technique augments previous suggestive selection techniques based on Gestalt theory [23, 29] by allowing the user to explicitly indicate what attributes she wants to search with.

Invert Filter Attribute

To exclude objects with a certain attribute from a selection, the user can invert the filter condition by performing a horizontal flip gesture on the filter card. In Boolean terms, that attribute will exist in an "*and not*" relationship with the other attributes in the set of filters. This expands the common "invert selection" command found in most drawing programs from spatial location to any other attribute type.

Compose and Reuse Selection with Filter Attributes

Performing a perfect selection is difficult and tedious. The key to an effective selection mechanism is to allow the user to compose selections with various techniques and reuse them in later tasks. Collection Objects enable users to freely combine and modify multiple attributes to achieve complex selections as well as reuse and modify existing COs without starting from an empty selection for every desired edit.

Imagine a user wants to select all the red petals within the top Easter egg (Figure 5a). She may simply hold a CO and tap to select the petals. While doing so, the system suggests possible attributes to her, such as the fill color and geometry (Figure 5a,b). Tapping the red fill color adds all the red petals into the CO, including ones in another egg. She can then copy the shape attribute of the top egg into the CO to constrain the selection within the specified boundary (Figure 5b,c). Then after changing colors , she can simply flip the boundary card switching to the red petals in another egg (Figure 5c,d).

In the floor plan example, the architect can also easily re-use the CO to select another set of lines with different stroke width by adjusting the value of the stroke attribute, which is being used as a search filter.

Compose Collection Object with Direct Manipulation

Objectifying selection allows users to physically manipulate Collection Objects directly, which further enables higher level composition of the scope of an existing selection.

Invert: Similar to flipping a filter attribute, to invert the search condition, existing CO can be horizontally flipped to transform to its inversion. Creating an empty CO and then flipping it allows a user to perform a *select all* operation.

Union, Intersection, and Subtraction: Holding two COs generates a new CO between the two. The user can then slide left or right to choose which Boolean operation he wants to use to generate this third new CO.

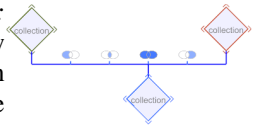

Figure 6 Boolean operation of COs

GROUPING WITH COLLECTION OBJECTS

In existing applications, a user can group selected objects to save the selection. However, this rigid structure also imposes difficulties for selecting objects within it. To select objects across groups, a user will have to first ungroup multiple times if they are maintained in a multi-level hierarchical structure. If she intends to save the group for later reuse, she will have to regroup them. Despite its rigid structure, grouping is still beneficial for forming meaningful structures. For example, a user can form a group of selected objects enabling operations such as scale, rotation, and alignment to be applied to the group instead of individual objects within the group. We seek to enable users to achieve such purposes while allowing quick navigation within the structure and preserving fast and fluid selection.

Creating a Group

A set of selected objects can be transformed into a group simply by flipping the CO vertically. Once the flip finishes, an icon appears within its diamond to indicate its grouped state. Flipping it again ungroups it. With CO, objects can still be freely added into or removed from the group using the selection techniques described above. By way of comparison, with existing applications, this has to be done either by ungrouping, reselecting, and then regrouping or using a layer panel to indirectly select objects by manually scrolling through a long list of items and visually matching them with graphical objects.

Hierarchical Structure vs. Lattice (Overlapping) Structure

Unlike the overlapping structure of normal Collection Objects, which is meant for high flexibility of selections, groups are organized in a tree structure. We choose the tree structure over the lattice structure (Figure 4 of [32]), as lattice grouping does not effectively reduce the number of objects to be managed and further a tree structure better matches users' understanding of the structure of a scene. These allow users to quickly navigate to the desired group with less mental effort. Contrarily, lattice structure, with which the flat groups are organized by most-recently-used order, requires the users to remember the sequence of when each group has been used [32].

Navigating the Tree Structure

A single tap of an object opens the Collection Object card of the group on the topmost level it belongs to (Figure 7a). If a user wants to access objects on the lower levels of the tree, she can hold the object she wants to navigate to and drag the diamond card (Figure 7). Immediately after her finger touches the diamond card, the levels associated with the held object appear underneath the finger, through which she can then drag the card vertically to navigate to different levels. While keeping her finger on the card, she can switch to hold another object to dive into a different branch of the tree.

Alternatively, she can hold the object and drag another finger vertically on the canvas to quickly navigate to the desired group. This allows her to rapidly navigate a multilevel tree structure with one hand in a local and bidirectional way, while adjusting the spatial attributes of the selected group with another hand.

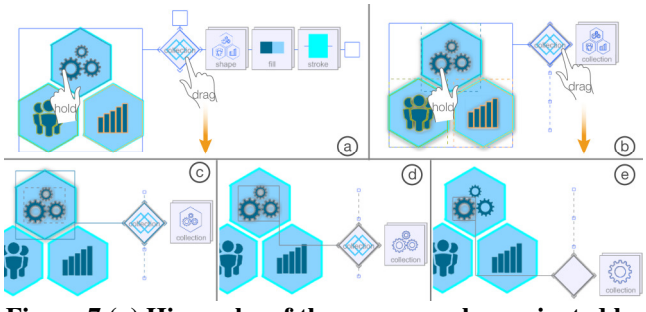

Figure 7 (a) Hierarchy of the group can be navigated by holding the object a user wants to navigate to and dragging the CO card vertically; (b) - (e) levels of held object updates as the user drags the card.

Selection of Objects Across Groups

One UI problem encountered when using grouping is that objects within a group or across groups cannot be easily selected. We solve this by keeping the group rigid to direct manipulation, easy to navigate, yet open to selection. When groups exist, they can be aggregated into a new CO by holding the CO and tapping on the groups. If a user wants to select objects within the group, she can touch her finger on the object she wants, drag down to navigate the level, as if it penetrates the structure, and release her finger at the level she wants to select the object. She can keep selecting objects in other groups simply by swiping down on the object as she becomes fluent with the gesture. Before releasing her finger to add a (sub-)group into the selection, she can swipe to the right, as in a marking menu, to add individual objects of the group into the selection rather than the group itself. Operations like alignment will be applied to the objects individually, instead of the group.

If a user flips this CO card, wishing to group an overlapping selection, a flip back animation will be played to indicate the current CO cannot be grouped because of the hierarchical structure.

ATTRIBUTES OF SELECTED OBJECTS

In addition to the filter attributes that define the selection, Collection Objects include an aggregation of the attributes present within the selected objects of the collection, similar to how a toolbar is updated to show common properties among selected objects in a WIMP UI. In this section, we describe how these Attribute Object cards may be manipulated to edit the selected objects.

Visual Representation

Given a collection of attributes, the UI displays all the values of attributes, similar to [16] but in a scalable way. For example, the stroke card shows strokes of different colors and widths. This representation is dynamically updated when attributes of the selected object are manipulated, Figure 9.

Showing Attributes: Iconic, Aggregates, or All

When a user first opens a Collection Object, an iconic representation of each of its attributes is shown on the Attribute Object cards, such as the color palette seen in Figure 8. The user is then able to expand the card to two levels. When the user first expands the card, an aggregate is shown. If the user continues to expand beyond this aggregate view, the complete set of values will appear.

Aggregates: When the user first expands the card with a vertical pinch gesture, an attribute-specific aggregate is shown, such as the gradient shown in Figure 8. For scalar values, such as x or y position, the max, min, and mean are shown. These aggregates allow the user to quickly find, copy, or change attribute values. As an example, if the user selects a group of objects and wishes to vertically align an unselected object with that group, she can quickly copy the "max" card from the "Y" attribute of the group to the new object; the object will immediately align with the top of the group. Alternatively, the user can also *replace* these cards.

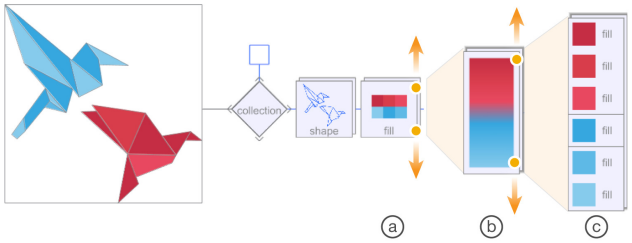

Figure 8. Example of three levels of expansion of attributes of objects in a collection: a) iconic; b) aggregate (in this case, a gradient), and c) a list of all values.

As an example, if she replaced the "min" Attribute Object card of the line thickness attribute with another value, all drawing objects whose line thickness was equal to the previous minimum would be updated.

All: If the user wishes to see all attribute values, she can continue to expand the card (Figure 8c). Additionally, each object within the collection will open its attribute object card in situ. This allows the user to copy or replace particular values within the set.

Direct Manipulation of Attribute Objects

In a traditional WIMP UI, the only possible operation on attributes of a selection of objects is assigning a single attribute value to all the selected objects. Conversely, with Attribute Objects, one can perform rich manipulation of attributes on selected objects.

In OOD [44], an attribute could be directly manipulated by holding its card and sliding another finger on the screen (e.g., holding the "drop shadow" card and dragging on the canvas translated the shadow to another position). We build on this by further using the landing object as a mode of touch input. Now, if the user holds the stroke attribute of the collection, and lands another finger on the canvas, moving her second finger will change the stroke width of all the objects in the collection. However, if she lands her finger on a particular object, only that object's stroke width will be changed. This allows users to quickly adjust the manipulation of an individual object or a collection of objects. Compared to the traditional form-filling paradigm and selection mechanism, this approach provides two important benefits. First, it allows the user to simultaneously manipulate attributes of several objects, without forcing them to change to the same value, as in traditional UIs. Second, the quasi-mode associated with the landing object can significantly reduce the need to switch between single selection and multi-selection. By doing so, one can fluidly conduct a sequence of actions without redoing complex selections or repeatedly grouping and ungrouping objects.

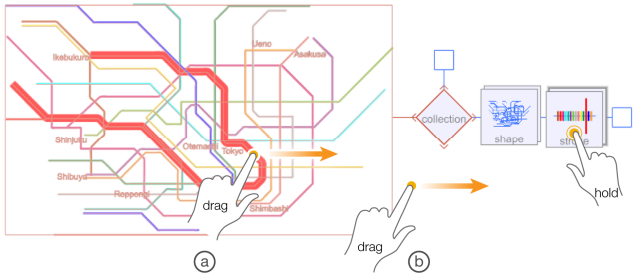

Figure 9. Example of direct manipulation of attributes. While holding the "stroke" card, the user can change the stroke thickness of a) an individual object within a collection by dragging their finger along the object or b) all thicknesses within the collection by dragging on the canvas.

Propagating Individual Values from the Collection

The traditional WIMP UI allows a user to set a value in a global widget to apply that attribute value to all the selected objects. However, even simple tasks using this functionality can require unreasonable effort. For example, to apply the fill color of one of the selected objects to all the others, the user will have to individually select the object, look up its fill attribute value, remember or copy the value, multi-select the other objects, and then set the value for the selection. With Collection Objects, this can be trivially achieved by holding the fill attribute of the collection of the objects and then tapping any other object to retrieve its fill color. Similarly, holding position attributes and tapping an object indicates he wants the whole collection aligned to it.

EXPERT EVALUATION

To validate whether Collection Objects is a useful replacement for the traditional way of working with multiple objects. We conducted an expert evaluation to gain feedback about the effectiveness and usefulness of Collection Objects. We were also interested in the utility of each individual interaction technique.

Apparatus

Our system is implemented as Win32 application in Windows 10 using OpenGL, running with 1920x1200px Wacom Cintq 24 HD display with capacitive finger touch and EMR pen hover + touch. Touch and stylus inputs can be simultaneously received and reliably differentiated.

Participants

We recruited 6 professional graphic designers (3 female, aged from 25 to 39) to evaluate our system. All participants have more than 5 years' experience using vector graphic tools, such as Adobe Illustrator and Inkscape. Participants were compensated $50 for a 90-minute session.

Procedure

Each expert review session consisted of the following stages:

Training on Collection Objects (25 - 30 minutes)

Participants were introduced to the concept of Collection Objects with a short verbal walkthrough of it techniques. Then, the experimenter trained the participant on the interface by having them complete a nine step drawing exploring all features of our system. During training, the experimenter described each interaction verbally to the participant and asked them to perform the actions. If the participant had difficulty performing the interaction, the experimenter would only then demonstrate it. We also kept each step of the drawing corresponding to each interaction technique simple, which allowed us to observe how the participants learned each technique and later applied them during the follow-up exercise part of the experiment.

Exercise and Freeform Usage (30 - 40 minutes).
Participants were then asked to practice the interaction techniques by replicating another drawing provided by the experimenter, this time without further guidance. After completing the tasks, participants were asked to explore the interface by creating their own illustrations.

Questionnaire and Interview (20 minutes)
After using the system, participants completed a questionnaire about Collection Objects. The questions addressed both the usefulness and usability of each technique using a 7-point Likert scale (1- Strongly disagree, 7- Strongly agree). Participants were then interviewed with open-ended questions to collect further feedback on utility and usability of Collection Objects and the workflow with COs, compared with traditional ways of working with multiple objects.

Results

Utility of Collection Objects
Participants were specifically asked to rate the usefulness of each of the interaction techniques. The results presented in Figure 10 show a common agreement among participants. This indicates that the interaction techniques enabled by Collection Objects are valuable and desired. As P3 commented: *The functions are fantastic. Especially with complex drawings, it can be so much more useful.*

Among the various techniques, participants strongly favored filtering by attributes, quick group navigation, and direct selection across groups (6/6 strongly agree), as they found these functions *powerful and unseen in other applications (P4).* Particularly, they support basic and frequently used functionalities, which are *extremely useful (P1) and set apart actual user experience (P1),* but *they (other applications) do fail on (P1).* Our tool *really has the potential (P6)* to help them achieve tasks that were previously tedious or *impossible (P6).*

Landing object as mode and sharing values to a CO were also acknowledged by participants (5/6 strongly agree, 1/6 agree), as they found *having so much control on what I select is really amazing. It makes the work process continuous and faster (P6).*

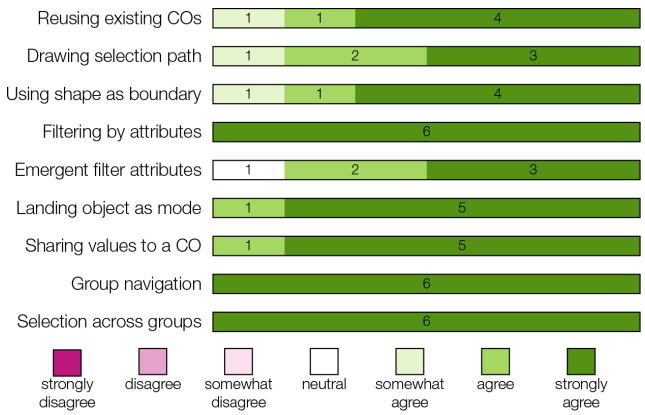

Figure 10. Likert-scale responses to "This technique was useful in my drawing tasks."

Other features also valued by participants included, the conventional selection using drawing paths, as well as the novel emergent filters only identified as *distracting* by P1.

Workflow vs Traditional Tools
Participants respond positively on the workflow with Collection Objects rating the statement "The experience of working with multiple objects in our system is fluid" as "strongly agree" (5/6) and "agree" (1/5). When interviewed, they attributed the fluidity to four features of our system:

1) less context switching with Collection Objects, since a series of selection, grouping, and manipulation of objects can be achieved through direct manipulation of one Collection Object and its two sets of attributes: *Having so many functionalities in this little thing [Collection Object] is great. You can just work with it* (*P2*), as opposed to the existing applications where the users *have to go to the toolbar, the layer panel, and a bunch of other stuff; they are all over the places (P2).*

2) composing and modifying selection with filter attributes, since it frees the user from having to *search the whole canvas by myself (themselves) (P4).* It also reduces mental effort as the users can rely on the search filters to select the unseen objects: *It's very relaxing. Many times I missed the things I wanted to select. They were either underneath other objects or I simply didn't see them. Sometimes I double checked my selection...(P4)* but *your system can reliably find them for me (P4).* Being able to modify the selection allows the users to easily switch to a related selection, which fits the general workflow. *Because the changes are kind of related. If you have just changed the opacity, it's very likely that you will continue changing it (P4).*

3) the open group structure and quick navigation of groups, which reduces the effort of managing the structure of the drawing: *with Illustrator, half the time, I am managing groups. That's not designing, that's just managing.* It also enables direct, quick, and flexible selection *without changing groups; the sliding to get to the group, man, it's so fast (P3).*

4) the rich manipulation of the aggregated attributes of the selected objects, which *is really designed with the considerations of supporting complex designs (P5).* It also allows for spontaneous manipulation of the selection: *sometimes (in other applications) you really just want to change it and quickly try out something, and you have to give up your selection. That slows down my design process (P2).*

Grouping with Collection Objects
Comparing against existing tools, all participants positively acknowledged how COs seamlessly combine overlapping selections with hierarchical group structures. They also valued our preference on using trees over lattice structures, as they found overlapping structures confusing (P2, 4).

P4. *If you cross group all over the place, then things belong to everything else, that's kind of weird. If you are preventing from doing that, that would be best. Because otherwise, that would actually be a mistake. But the ease of selection within the groups is really important.*

Being able to easily change the scope of the group was found useful for maintaining a cleaner and flatter group structure, It reduces *unnecessary groups (P1), which are a nightmare for later changes (P1),* but usually created *to avoid the ungroup, select, and regroup procedure (P1).*

Features Suggested by Participants

Once the participants understood the concept, they started to explore the limits of our system and discovered features that had not been demonstrated. As P2 noted, "*The features are so rich. You see I was doing a lot more than your system has, that's because I understand it"*.

Interesting behaviors and features were observed or suggested by our participants. After learning that the shape of an existing object could be used as the boundary of a selection, P1 tried to share the shape of the canvas to achieve a select all function. Having successfully done that, P1 then tried to select all the red objects on the top layer, by sharing the shape of the layer object. However, since our system does not provide a shape attribute for layer object, P1 then tried to share the whole layer object with a CO. Although this is not supported by the current system, it shows that the participant understood the concept that selection can be composed by setting up a boundary and then filtering it. Since the layer object is itself a container, it also reveals that existing COs could be used to define the scope of a new collection, further expanding the expressiveness of Collection Objects.

Learning Curve

Participants agreed that the techniques were easy to learn (5/6 strongly agree, 1/6 agree). We found them could learn the few simple rules which formed the basis of all interactions: top of the diamond is for searching and the right is for editing, holding an object indicates the interaction is bounded to it, and attributes are objects. We intentionally repeated these principles in the training, as we learned this is an effective way to teach how to use the system. Three of our participants also took part in study of [44] and they found the new interface uniform consistent with the original concept (3/3 strongly agree), as P1 commented: *it's like you have learned how to use a game controller, and then you can play all the games.*

Summary

The results of the expert evaluation suggest that Collection Object enables rich, advanced, and fluid manipulation with multiple objects, which were previously tedious or even impossible in existing applications.

DISCUSSION AND FUTURE WORK

While participants positively acknowledged the usefulness of Collection Objects, they also identified usability problems that can be addressed by future work. P1 found that the richness of features in our system is "overwhelming". As P1 noted, "*it has so many features, and sometimes I discovered something you hadn't shown me. I was worried there are some things in the system I don't know, and I may accidentally activate them.*" This indicates that the usability can be improved by clearly communicating the capability of the system to the users. P6 found that having COs floating made the working area messier. Although, P4 noted the opposite as she found it useful to have them on the canvas for immediate access to previous selections, a future system can provide a side panel to organize Collection Objects.

One drawback of the direct manipulation paradigm, is selecting unseen objects, for example, objects may be simply out of the view, in a complex scene, or covered by other objects [13]. Bringing unseen objects into view requires extensive manual searching and visual comparisons [13]. Our select-by-search functionalities mitigate this problem, e.g., filters within one CO can be reversed individually, they can be chained together with an AND relationship, and OR relationship is partially supported within each attribute. To extend such features to different attributes, the chain metaphor can be augmented to allow each node of the chain to host more than one card and several chains to be parallel. As such, a search task can be done by spatially manipulating the attribute cards. It would also be interesting to explore whether this could achieve a regex complete complexity.

Future work can also explore the inclusion of other selection techniques into Collection Objects. For example, instead of using specific style attributes as search filters, Gestalt principles can be directly used as selection filters, with their weights accessible and adjustable through the fuzziness attribute. Potential selection results can be suggested directly to the user in a way similar to the emergent filter attributes.

One clear future step is to compare with existing WIMP tools. This may reveal the different strategies users may take to achieve a certain goal and the difference of performance, such as the amount of steps required.

The OOD system provides a micro view of objects by allowing the user to freely disassemble objects into smaller units – Attribute Objects. Complementary to it, Collection Objects explores the macro manipulation of a collection of objects, which provides the opportunity to explore interaction at a higher level. creating a large number of objects is itself difficult and time-consuming. Future work could explore how Collection Objects can enable easy and fast generation of multiple objects of certain patterns. This could be useful in the area of computer-aided design, data visualization [40], and texture-based drawing [18].

CONCLUSION

We have presented Collection Objects, the embodiment of the transient selection and the rigid group structure. The physical embodiment combined with direct physical manipulation gestures demonstrates the rich properties of Collection Objects, which not only enables a single quick selection-action operation, but also provides a coherent mechanism wherein a series of manipulation of multiple objects can be rapidly performed with no context switching. Demonstrated through expert evaluation, Collection Objects enables quick, flexible, fluid interaction with multiple objects. We hope Collection Objects will enable complex applications with direct physical input.

REFERENCES

1. Johnny Accot and Shumin Zhai. 1997. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (CHI '97). ACM, New York, NY, USA, 295-302. DOI=http://dx.doi.org/10.1145/258549.258760
2. Robert St. Amant and Thomas E. Horton. 2002. Characterizing tool use in an interactive drawing environment. In *Proceedings of the 2nd international symposium on Smart graphics*(SMARTGRAPH '02). ACM, New York, NY, USA, 86-93. DOI=http://dx.doi.org/10.1145/569005.569018
3. Michel Beaudouin-Lafon. 2000. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (CHI '00). ACM, New York, NY, USA, 446-453. DOI=http://dx.doi.org/10.1145/332040.332473
4. Benjamin B. Bederson, James D. Hollan, Allison Druin, Jason Stewart, David Rogers, and David Proft. 1996. Local tools: an alternative to tool palettes. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (UIST '96). ACM, New York, NY, USA, 169-170. DOI=http://dx.doi.org/10.1145/237091.237116
5. Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: modeling finger touch with fitts' law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13). ACM, New York, NY, USA, 1363-1372. DOI: http://dx.doi.org/10.1145/2470654.2466180
6. Eric A. Bier and Steven Freeman. 1991. MMM: a user interface architecture for shared editors on a single screen. In *Proceedings of the 4th annual ACM symposium on User interface software and technology* (UIST '91). ACM, New York, NY, USA, 79-86. DOI=http://dx.doi.org/10.1145/120782.120791
7. Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. 1993. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '93). ACM, New York, NY, USA, 73-80. DOI=http://dx.doi.org/10.1145/166117.166126
8. Sebastian Boring, Marko Jurmu, and Andreas Butz. 2009. Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7* (OZCHI '09). ACM, New York, NY, USA, 161-168. DOI=http://dx.doi.org/10.1145/1738826.1738853
9. Peter Brandl, Clifton Forlines, Daniel Wigdor, Michael Haller, and Chia Shen. 2008. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In*Proceedings of the working conference on Advanced visual interfaces* (AVI '08). ACM, New York, NY, USA, 154-161. DOI=http://dx.doi.org/10.1145/1385569.1385595
10. Bill Buxton. Chunking and phrasing and the design of human-computer dialogues. *IFIP '86*, 475–480.
11. Fanny Chevalier, Pierre Dragicevic, and Christophe Hurter. 2012. Histomages: fully synchronized views for image editing. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST '12). ACM, New York, NY, USA, 281-286. DOI=http://dx.doi.org/10.1145/2380116.2380152
12. Paul M Fitts. (1954). The information capacity of the human motor system in controlling the amplitude of movement.*Journal of experimental psychology*,*47*(6), 381.
13. David M Frohlich. The history and future of direct manipulation. *Behaviour & Information Technology* 12.6 (1993): 315-329. .
14. Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, and Francois Guimbretiere. 2005. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, USA, 451-460. DOI=http://dx.doi.org/10.1145/1054972.1055035
15. Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. 2010. Pen + touch = new tools. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10). ACM, New York, NY, USA, 27-36. DOI=http://dx.doi.org/10.1145/1866029.1866036
16. Raphaël Hoarau and Stéphane Conversy. 2012. Augmenting the scope of interactions with implicit and explicit graphical structures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). ACM, New York, NY, USA, 1937-1946. DOI=http://dx.doi.org/10.1145/2207676.2208337
17. Steve Jobs. Mac World Expo 2007: Keynote.
18. Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 351-360. DOI: http://dx.doi.org/10.1145/2556288.2556987
19. Bum chul Kwon, Waqas Javed, Niklas Elmqvist, and Ji Soo Yi. 2011. Direct manipulation through surrogate objects. In *Proceedings of the SIGCHI Conference on*

Human Factors in Computing Systems (CHI '11). ACM, New York, NY, USA, 627-636. DOI=http://dx.doi.org/10.1145/1978942.1979033

20. David Kurlander and Eric A. Bier. 1988. Graphical search and replace. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (SIGGRAPH '88), Richard J. Beach (Ed.). ACM, New York, NY, USA, 113-120. DOI=http://dx.doi.org/10.1145/54852.378495
21. Edward Lank and Eric Saund. 2005. Sloppy selection: Providing an accurate interpretation of imprecise selection gestures. *Comput. Graph.* 29, 4 (August 2005), 490-500. DOI=http://dx.doi.org/10.1016/j.cag.2005.05.003
22. Jakob Leitner and Michael Haller. 2011. Harpoon selection: efficient selections for ungrouped content on large pen-based surfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 593-602. DOI=http://dx.doi.org/10.1145/2047196.2047275
23. David Lindlbauer, Michael Haller, Mark Hancock, Stacey D. Scott, and Wolfgang Stuerzlinger. 2013. Perceptual grouping: selection assistance for digital sketching. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces* (ITS '13). ACM, New York, NY, USA, 51-60. DOI=http://dx.doi.org/10.1145/2512349.2512801
24. Yuexing Luo and Daniel Vogel. 2014. Crossing-based selection with direct touch input. In*Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 2627-2636. DOI: http://dx.doi.org/10.1145/2556288.2557397
25. I. Scott MacKenzie, Abigail Sellen, and William A. S. Buxton. 1991. A comparison of input devices in element pointing and dragging tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '91), Scott P. Robertson, Gary M. Olson, and Judith S. Olson (Eds.). ACM, New York, NY, USA, 161-166. DOI: http://dx.doi.org/10.1145/108844.108868
26. Wendy E. Mackay. 2002. Which interaction technique works when?: floating palettes, marking menus and toolglasses support different task strategies. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (AVI '02), Maria De Marsico, Stefano Levialdi, and Emanuele Panizzi (Eds.). ACM, New York, NY, USA, 203-208. DOI=http://dx.doi.org/10.1145/1556262.1556294
27. Thomas P. Moran, Patrick Chiu, and William van Melle. 1997. Pen-based interaction techniques for organizing material on an electronic whiteboard. In *Proceedings of the 10th annual ACM symposium on User interface software and technology* (UIST '97). ACM, New York, NY, USA, 45-54. DOI=http://dx.doi.org/10.1145/263407.263508
28. Ken Perlin and David Fox. 1993. Pad: an alternative approach to the computer interface. In*Proceedings of the 20th annual conference on Computer graphics and interactive techniques*(SIGGRAPH '93). ACM, New York, NY, USA, 57-64. DOI=http://dx.doi.org/10.1145/166117.166125
29. Florian Perteneder, Martin Bresler, Eva-Maria Grossauer, Joanne Leong, and Michael Haller. 2015. cLuster: Smart Clustering of Free-Hand Sketches on Large Interactive Surfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (UIST '15). ACM, New York, NY, USA, 37-46. DOI: http://dx.doi.org/10.1145/2807442.2807455
30. Jef Raskin. The Humane Interface: New Directions for Designing Interactive Systems. 2000.
31. Chun Russel. Adobe Flash Professional CC Classroom in a Book, Adobe Press. 2014.
32. Eric Saund, David Fleet, Daniel Larner, and James Mahoney. 2003. Perceptually-supported image editing of text and graphics. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (UIST '03). ACM, New York, NY, USA, 183-192. DOI=http://dx.doi.org/10.1145/964696.964717
33. Ben Shneiderman. The future of interactive systems and the emergence of direct manipulation ." *Behaviour & Information Technology* 1.3 (1982): 237-256.
34. Katie A. Siek, Yvonne Rogers, and Kay H. Connelly. 2005. Fat finger worries: how older and younger users physically interact with PDAs. In *Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction* (INTERACT'05), Maria Francesca Costabile and Fabio Paternò (Eds.). Springer-Verlag, Berlin, Heidelberg, 267-280. DOI=http://dx.doi.org/10.1007/11555261_24
35. Randall B. Smith. 1986. Experiences with the alternate reality kit: an example of the tension between literalism and magic. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface* (CHI '87), John M. Carroll and Peter P. Tanner (Eds.). ACM, New York, NY, USA, 61-67. DOI=http://dx.doi.org/10.1145/29933.30861
36. Sven Strothoff, Wolfgang Stuerzlinger, and Klaus Hinrichs. 2015. Pins 'n' Touches: An Interface for Tagging and Editing Complex Groups. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (ITS '15). ACM, New York, NY, USA, 191-200. DOI: https://doi.org/10.1145/2817721.2817731
37. Ivan E. Sutherland. 1964. Sketch pad a man-machine graphical communication system. In*Proceedings of the SHARE design automation workshop* (DAC '64). ACM,

New York, NY, USA, 6.329-6.346. DOI=http://dx.doi.org/10.1145/800265.810742

38. Michael Terry and Elizabeth D. Mynatt. 2002. Side views: persistent, on-demand previews for open-ended tasks. In *Proceedings of the 15th annual ACM symposium on User interface software and technology* (UIST '02). ACM, New York, NY, USA, 71-80. DOI=http://dx.doi.org/10.1145/571985.571996
39. Edward Tse, Chia Shen, Saul Greenberg, and Clifton Forlines. 2006. Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In *Proceedings of the working conference on Advanced visual interfaces* (AVI '06). ACM, New York, NY, USA, 336-343. DOI=http://dx.doi.org/10.1145/1133265.1133336
40. Bret Victor. Drawing Dynamic Visualization. 2013.
41. Daniel Wigdor. 57.4 :*Invited Paper*: The Breadth—Depth Dichotomy: Opportunities and Crises in Expanding Sensing Capabilities. SID Symposium Digest of Technical Papers, 42: 845–848.
42. Daniel Wigdor, Hrvoje Benko, John Pella, Jarrod Lombardo, and Sarah Williams. 2011. Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In*Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '11). ACM, New York, NY, USA, 1581-1590. DOI=http://dx.doi.org/10.1145/1978942.1979173
43. Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 1083-1092. DOI=http://dx.doi.org/10.1145/1518701.1518866
44. Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-Oriented Drawing. In*Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (CHI '16). ACM, New York, NY, USA, 4610-4621. DOI: http://dx.doi.org/10.1145/2858036.2858075
45. Pengfei Xu, Hongbo Fu, Chiew-Lan Tai, and Takeo Igarashi. 2015. GACA: Group-Aware Command-based Arrangement of Graphic Elements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 2787-2795. DOI: http://dx.doi.org/10.1145/2702123.2702198
46. Pengfei Xu, Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai. 2012. Lazy selection: a scribble-based tool for smart shape elements selection. *ACM Trans. Graph.* 31, 6, Article 142 (November 2012), 9 pages. DOI=http://dx.doi.org/10.1145/2366145.2366161