Tumble! Splat! Helping Users Access and Manipulate Occluded Content in 2D Drawings

Gonzalo Ramos[1], George Robertson[2], Mary Czerwinski[2], Desney Tan[2], Patrick Baudisch[2], Ken Hinckley[2], and Maneesh Agrawala[3]

[1]Univ. of Toronto
Department of Computer Science
10 King's College Road, Room 3302
Toronto, Ontario M5S 3G4, Canada

bonzo@dgp.toronto.edu

[2]Microsoft Research
One Microsoft Way
Redmond, WA 98102, USA

{ggr, marycz, desney, baudisch, kenh}@microsoft.com

[3]Univ. of California, Berkeley
615 Soda Hall, Mail Code #1776
Berkley, CA 94720, USA

maneesh@cs.berkeley.edu

ABSTRACT
Accessing and manipulating occluded content in layered 2D drawings can be difficult. This paper characterizes a design space of techniques that facilitate access to occluded content. In addition, we introduce two new tools, Tumbler and Splatter, which represent unexplored areas of the design space. Finally, we present results of a study that contrasts these two tools against the traditional scene index used in most drawing applications. Results show that Splatter is comparable to and can be better than the scene index. Our findings allow us to understand the inherent design tradeoffs, and to identify areas for further improvement.

Categories and Subject Descriptors
H5.2 [**Information interfaces and presentation**]: User Interfaces – Graphical user interfaces.

Keywords
Layer management, 2D drawing, occlusion, interaction technique.

1. INTRODUCTION
Users often need to edit or interact with compositions that contain overlapping 2D objects. However, selecting and manipulating objects can be difficult when they are occluded by other objects. Traditionally, there have been three approaches to accessing occluded 2D objects: 1) using direct manipulation to move overlapping objects until the desired object becomes easily selectable; 2) using a widget such as a context menu to cycle through the objects that are below or above a particular point in the scene; or 3) using a modeless scene index, an ordered linear list of thumbnails, to directly access all objects within the scene. While widespread in most drawing applications, we assert that these approaches do not scale well when the total number of objects in the scene grows large, or when the scene contains elements whose thumbnails are visually similar. For example, the first two approaches become tedious even for a moderate number of elements. Similarly, space requirements of linear lists can grow larger than the screen that contains them, making the task of finding a single object difficult. Furthermore, the thumbnails in such lists do not always preserve all the objects' original visual attributes such as shape, color, size, or position.

We believe there is an absence of a systematic exploration of the implications of manipulating the design parameters of the existing techniques. For example, scene indexes such as those in applications like Adobe Illustrator only preserve an object's shape, color and drawing order (i.e. layer order), making it difficult to distinguish between objects with similar drawing styles that only differ in size. In contrast, we believe that preserving more or different visual attributes may have the potential to overcome several limitations of current techniques for accessing and manipulating overlapping 2D drawings. Ultimately, we argue that there is inadequate support for manipulating occluding 2D content in applications for novice users. Such applications (e.g. PowerPoint) mostly rely on sequential access to occluded content through context menus and can benefit from both revision of existing tools and exploration of new solutions.

In this paper, we use objects' visual attributes to suggest a design space for techniques that help users access and manipulate occluding 2D drawings. Building upon this space, we introduce *Tumbler* and *Splatter*, two tools that instantiate unexplored areas of this design space. We then describe a user study that investigates these techniques' performance and compares them to a standard scene index. Results provide better understanding of the tradeoffs as well as usability issues of these interfaces when accessing and manipulating occluding 2D content. We present improvements to the designs of the new and existing tools in order to make users more effective when interacting with occluded 2D content. Finally, we discuss the implications of our findings and lay out directions for future research.

2. RELATED WORK
Occlusion is often found in volumetric data, graphs and virtual models. McGuffin *et al.* [10] identify 3 main strategies used to alleviate occlusion problems in 3D: transparency filters, cutting geometries, and spatial transforms. They also present several deformation methods for exploring volumetric data. Users apply these methods through a series of in-place manipulations that separate and reveal regions of interest in a 3D volume. While exploratory, these techniques have potential within a 2D editing workflow. Carpendale et al. [6] and Sonnet et al. [14] also use spatial transforms to scale or move elements along a user-defined line of sight, thus providing access to areas that would be otherwise occluded. To our knowledge, these deformation techniques have not been formally evaluated.

Occlusion on window managers has been addressed by commercial products and researchers alike. Apple's *exposé* [1] uses a spatial transformation that makes occluding objects accessible. But this transform conveys neither layering information about overlapping windows nor the means to alter their order. Robertson's *target chooser* [12] allows users to select occluded or distant windows by casting a ray from the pointer's position and manipulating it as the pointer moves. Still, this technique relies on the user explicitly discovering completely occluded windows. Beaudouin-Lafon [4] also studies overlapping windows and proposes browsing through a set of (occluded) tabbed windows as if they were cards in a Rolodex. He also presents the concept of peeling a pile of windows as if they were sheets of physical paper. This peeling allows users to reveal hidden windows. Dragicevic [7] extends these ideas to allow drag-and-drop between overlapping windows. While compelling, all these techniques require users to tediously search for hidden objects by moving occluders one at a time.

Baudisch et al. [3] introduce multiblending, a framework of transparency filters that allow for the simultaneous display of overlapping windows. While user studies reveal how multiblending improved readability, it is not clear how it can facilitate access to occluded content. Ishak et al. [9] use transparency filters in their context-aware free-space transparency (FST) to reveal hidden content in window managers and facilitate users' interaction with that content. They present 3 interaction techniques that capitalize on FST. *Pop-through* lets users pierce through an occluding window onto windows directly underneath by adjusting the pressure a user applies with stylus. The *focus filter* acts as a magic lens [5] that reveals hidden content underneath a particular area; and *mouse-over pie menus* provide a selector that lets users choose from the windows underneath a particular pixel. While promising, FST relies on users probing the right spot on the screen in order to be effective. These 3 techniques seem well suited to be used in concert with each other, but we could not find usability evaluations on FST.

In summary, there are a number of tools seeking to facilitate the access and manipulation of occluded content. However, we found no systematic attempt to characterize the design space of such tools. In addition, to our knowledge, there is no empirical data for many of these tools.

3. TOOLS FOR OCCLUDED DRAWINGS: A DESIGN SPACE

Overlapping 2D drawings can be thought of as individual objects, which exist in infinitely thin layers stacked on top of each other. We can then assign each object a z-value that corresponds to its location in the stack. Although unoccluded objects are easily accessible using direct manipulation, accessing an occluded object using direct manipulation becomes difficult. Exploded views of an occlusion group are commonly used to regain access to particular objects of interest. However, exploded views usually incur a cost as they may discard visual attributes and spatial relationships between objects – e.g., shape, size, position, or z-value. In this section we analyze these different attributes and their relationships and use them as the cornerstones of a design space that help us characterize existing as well as new tools.

3.1 Design Cornerstones

We define 4 cornerstones describing visual attributes and spatial relationships of the objects within a scene: *z-value*, *shape*, *size*, and *location*. Each of these cornerstones helps us with identifying and differentiating objects, as well as selecting and manipulating them. We use parallel coordinates [8] to visualize the four dimensions of this design space, where each coordinate value indicates how much of an object's attribute a tool preserves. Is not easy to quantify how much a tool preserves an object's attributes. However, we only require a (continuous) magnitude that can be compared – e.g., tool *A* preserves *shape* more than tool *B*. Figure 1 illustrates how a scene index or *Palette* is plotted as a line that connects a high (+++) *z-value* with a moderate (++) *shape* and null (-) *size* and *location*.

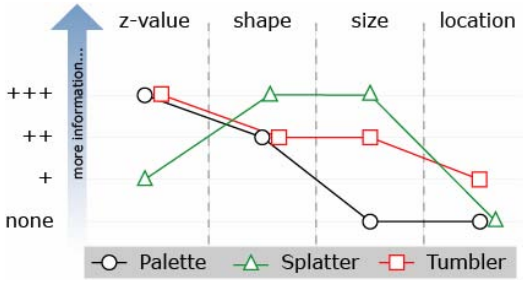

Figure 1: Parallel plots represent different designs. E.g., Palette does not preserve size or location. Tumbler and Splatter probe different regions of the design space.

Regions of this space corresponding to tools that keep most of an object's attributes intact seem attractive. To achieve this, such tools must rely on mainly adjusting objects' transparency or rendering style – i.e., showing an object as its outline. However, early pilot studies showed us that such visualization approaches have problems including ambiguity, scalability, and occlusion. Instead, we explore different regions of this design space by presenting two new in-place tools to facilitate the access and manipulation of overlapping 2D drawings: *Tumbler* and *Splatter*.

4. TUMBLER

Tumbler is a tool designed for novice users to operate on occluded 2D objects by presenting an in-place 3D stacked view of the occlusion group and their respective layers (Figure). Each layer in Tumbler is represented by a frame with a distinctive color and has at least one object. Each layer's frame has icons on its edges (Figure b) that show what and how many objects a layer has as well as their relative position. Each icon is aligned with its object's center of mass and matches the object's drawing style. Each layer also contains a region where widgets can be added to expand functionality – e.g., the *eye switch* in Figure b.

Users tumble a group of objects by holding down a modifier key and dragging their pointing device diagonally from a group of objects. Also, users can change the separation between Tumbler's frames by dragging the pointer nearer to or further from the stack's bottom, along Tumbler's current orientation if the modifier key stays pressed. While active, Tumbler replaces the group of objects it came from. We change the rendering style of objects outside of Tumbler by making them look sketch-like – e.g., by applying an edge-altering filter. This change in rendering style differentiates objects from their original depiction, and also makes all objects in the drawing canvas visible, thus providing a virtual x-ray view of the whole canvas.

Users select objects in Tumbler by clicking on an empty area of the layer which contains them. Alternatively, users can cycle through and select Tumbler*'s* layers by using the mouse wheel or arrow keys. Users browse Tumbler by hovering over it with their

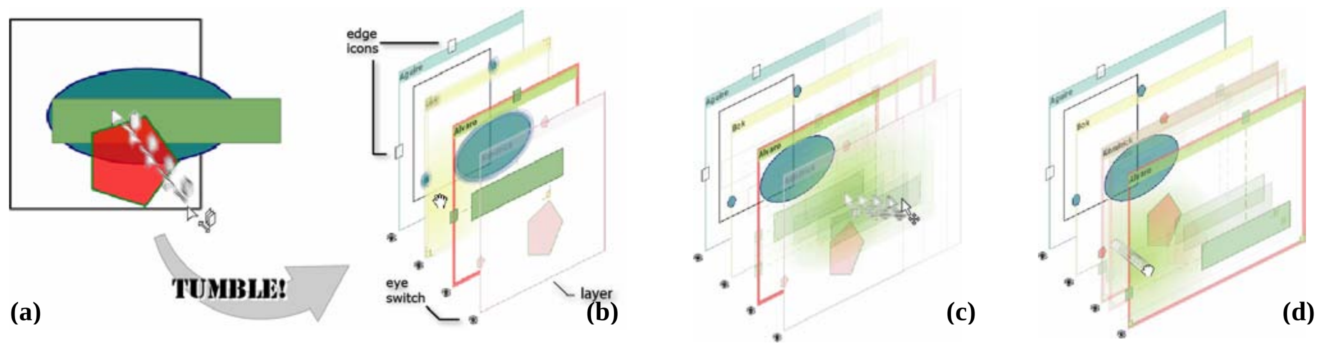

Figure 2: Tumbler's ecology: a) An occlusion group gets expanded into a Tumbler. b) Layer's elements (the layer below the cursor is blurred to denote how it "shakes"). c) Changing an object's x and y position. d) Layering - i.e., changing a layer's z-value.

pointer. We provide two cues that indicate what layer the pointer is over. First, when the mouse pointer enters a layer, the layer shakes to make it salient. This is a form of *kinetic visualization* [16]. Second, we tint the layer's surface under the pointer with a soft spotlight in the layer's own color (Figure b-d).

Users can change the z-order of layers by dragging layers within Tumbler's stack (Figure c). Tumbler also lets users adjust the x or y position of the selected object by dragging it within that layer. When doing this, the object being moved casts a shadow on the underlying layers to assist in alignmet tasks. Tumbler is closed by collapsing its layers, or by clicking outside its layers.

4.1 Design Considerations

Tumbler preserves the z-value, relative shape, relative size, and relative position of the objects it represents (Figure 1). Unlike traditional scene indexes that list all objects in a drawing, Tumbler provides an alternate visual representation of a group of objects. This group can be defined by the user or by the tool. If the user defines a group of objects, Tumbler includes the objects in that group. If no group is defined, Tumbler uses an object intersection test to determine what objects to include in its stack. The tool presents all the objects that intersect the particular object on which the tool was activated.

Tumbler uses an axonometric view for the stack of layers. The advantage of this representation over a 3D perspective is that axonometry allows users to compare the relative size of objects at different depths. One disadvantage of axonometric representations is that it may lead to a perceptual illusion known as the Necker Cube effect, in which it is difficult to discern the front and bottom sides of a cube. We alleviate this effect using hidden-line techniques that reinforce the layers' true order. We also adjust the opacity of the layers to provide a clearer view of the selected one.

Tumbler provides animated transitions between its different states and orientations to show correspondence of objects at different states. Changing the tool's orientation provides information that can help users to discern a layer or object's z-value. As objects in Tumbler shift from one point of view to the next, visual parallax effects reveal information about the stack structure. Nonetheless, these animations, combined with manually adjusting the separation between layers, impact the tool's performance time-wise.

5. SPLATTER

Splatter (Figure) is a tool designed for novice users to operate on a group of occluding 2D objects. This tool temporarily separates the occluding group's objects and makes them accessible by direct manipulation. Users create a Splatter by holding a modifier key and clicking on a group of objects. This brings up an exploded view (Figure a) of scaled proxies of the objects originally overlapping the one that the user clicked on.

While Splatter is active, the original objects remain visible at their original location, but with a different rendering style. Like Tumbler, Splatter uses a sketch-like rendering style for this purpose. The rest of the objects in the scene are faded to a subtle alpha level that provides a virtual x-ray view onto the whole canvas. Splatter uses a force directed layout algorithm where each participating object is initially scaled down and put along the ray that contains the object's center of mass and the tool's activation point, at a distance proportional to the object's size. This layout roughly preserves objects' spatial relationships.

Proxies are connected to the point users clicked on with a beam that has a distinctive color, similar to the ones used in [2, 11, 13]. When the pointer's cursor passes above a proxy or its beam, the original object in the scene becomes highlighted by going back to its original rendering style. Users can then click on the proxy if they want to select the object. Users can rearrange the tool's layout by dragging its proxies. This allows them to inspect z-value relationships by looking at whether a beam does or does not occlude another proxy or beam.

Splatter lets users adjust the z-value of an object within the group (Figure c): If object A is occluded by object B, a user can drag A's proxy over B's to put A directly on top of B. If A is on top of B, a user can drag A's proxy over B's while holding a modifier key to put A directly under B. Users release the mouse button while in the desired state to commit the (un)adjusted z-value.

5.1 Design Considerations

Splatter's original implementation preserves the shape, relative size, and to some degree the z-value and relative position of the objects it represents (Figure 1). While the tool's exploded view of objects' proxies does not fully convey all location information, it exposes all the objects in a group as directly selectable targets. Splatter uses animated transitions when it is activated and deactivated to show correspondence between objects and its proxies. In the current implementation, these animations introduce a 300 ms activation overhead.

6. BASELINE SOLUTION: THE LAYERS PALETTE

While the layers Palette is the predominant tool for operating on overlapping 2D layers for professional applications, that is not the case for office-style ones. Nonetheless we take a conservative stand and use is as a baseline tool. The Palette is an ordered list

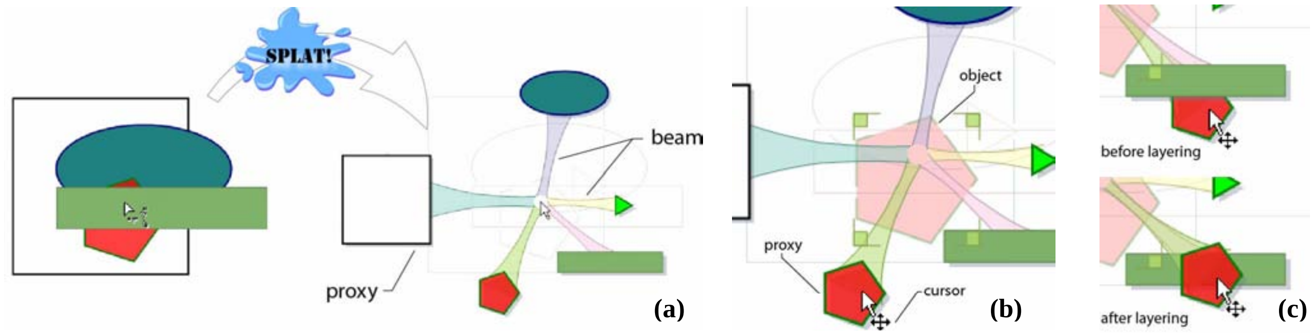

Figure 3: Splatter ecology. a) An occlusion group gets splatered. b) Hovering over an object's proxy reveals the objects true attributes. c) Layering: as the red pentagon is dragged close to the green rectangle, as the cursor enters the rectangle the pentagon will be put directly on top of the ellipse.

that displays all objects in a drawing as tiles vertically arranged inside a rectangular frame (Figure 4). The vertical order of these tiles corresponds with the z-value of the drawing's objects.

Each tile generally consists of three components: a thumbnail of the object it represents, an eye or visibility switch, and a label. An object is selected by clicking on either its corresponding label or thumbnail. The selection is also indicated by highlighting the corresponding tile. An object can be temporarily hidden in a drawing by clicking on its corresponding eye switch. Clicking the switch again shows the object again. In our instantiation of this tool, Alt-clicking on the switch hides all objects but the one the switch corresponds to. The Palette can be used to adjust the z-value of an object by clicking and dragging its tile to a different position in the list. If there are more objects than lines that fit on the Palette, a scroll bar is displayed.

6.1 Design Considerations

Each thumbnail in the Palette is an image of the object it represents, scaled to fill the thumbnail's boundaries (20x20 pixels). This behavior is standard in current vector-drawing applications, since preserving objects' relative size may render small objects invisible in the presence of large ones. The Palette provides at-a-glance information about objects' z-values and seems well suited for tasks that involve adjusting objects' z-values. However, the tool is likely to degrade as the number of objects in the scene grows. Because similar objects with different sizes are indistinguishable by just in the *thumbnail* views, using the Palette alone may not be effective for searching or selection tasks.

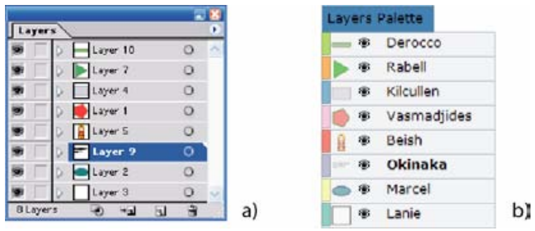

Figure 4: Layers Palette. a) Adobe Illustrator's. b) The one used in our study

7. USABILITY STUDY 1

While Splatter seems well suited for quick selection of occluded drawings, Tumbler shows promise for tasks that involve changing an object's z-value. At the same time the Palette brings the familiarity and simplicity of linear lists. These 3 tools complement each other well by giving users a number of options to access occluded drawings either from a global (Palette) or local (Tumbler, Splatter) perspective. These tools also let users perform tasks such as object selection or layer manipulation in different ways, which may be correspond better to a user's particular skills. We conducted a study to test the effectiveness of Tumbler and Splatter as well as to gather formative usability data particular to each tool. We were also interested in comparing the different tools with one another and in identifying their respective advantages and disadvantages.

7.1 Participants

18 participants (9 female), aged 21 through 53, participated in this study. All were familiar with at least one 2D vector drawing application – e.g., PowerPoint, Adobe Illustrator, MS Paint. Participants received software gratuities.

7.2 Apparatus

We ran the study on a 2.8 GHz P4 PC with 2GB of RAM running Windows XP Pro. We used two NEC 1880SX LCD panels running at a resolution of 1280x1024.

7.3 Tasks and Stimuli

The study consisted of two tasks: selection and layering. Each task used two displays: primary (D1) and secondary (D2). For all tasks, D1 always contained the scene participants interacted with: different geometrical shapes arranged in one or more occlusion groups.

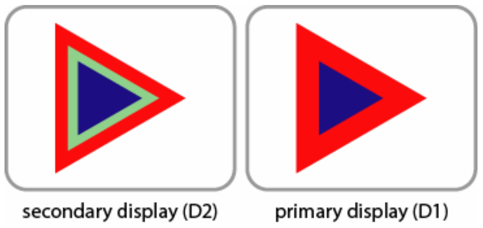

Figure 5: Layering: Target scene appears on the secondary display while trial scene appears on the primary one. The user has to adjust only the z-value of the (hidden) green triangle so that the primary and secondary displays match.

In each selection trial, a target object was shown on D2 and participants had to find and select this target within a scene on the D1. There was always *exactly one* possible target object in each scene, chosen from a set of 8 objects: A (small or medium) (red or green) (triangle or hexagon). A user-selected object in the scene was highlighted with a rectangular marquee.

In the layering task, users had to adjust the z-value of a particular object contained in a drawing shown on D1. In particular, participants had to adjust the z-value of a hidden object so that it lay between two target objects. The target objects were always overlapping and totally or partially visible. The stimulus for this task (e.g. Figure 5) was presented in D2 and was identical to the one shown on D1, but with the hidden object having its correct z-value – i.e., partially visible and between the target objects.

For a given layering task the hidden and target objects had the same shape (triangle or hexagon), and different, randomly chosen colors (red, green, and blue) and sizes (small, medium, and large). The hidden object was always at some z-distance underneath both target objects. For each of these tasks, scenes contained 4 types of objects: rectangles, ellipses, pentagons, and bitmaps. In each scene we added a set of distractor objects so that users had to search for the target(s) in a busy scene. For the selection tasks, distractors had the same shape and color as the target, but a different size. For the layering task, distractors had the same shape as the targets and were red, green or blue. A distractor always had a difference size than the target, but the same shape and color. We varied the number of distractors to manipulate the scene's complexity: 1% and 5% of the objects were distractors in *low* and *high* complexity scenes, respectively. Distractors were uniformly located across a scene's occlusion groups. While Tumbler and Splatter are not search tools, we control a scene's complexity so as not to bias the study against the Palette condition.

We use [15] as the basis for defining the *Complexity* of a scene. For our study, *Complexity* is a function of the number of occlusion groups (NG) and the number of objects in an occlusion group (NOG). In particular, a scene has *low* complexity if 0<NG<4 and 3<NOG<6. Conversely, a scene has *high* complexity if 3<NG<7 and 3<NOG<8.

7.4 Procedure and Design

We used a 3 tool (*Palette, Splatter, Tumbler*) × 2 task (*Selection, Layering*) × 2 complexity (*low*, *high*) within-subjects design. The dependent variables were Trial Time and Mistrials. Trial Time is the time elapsed between the moment a participant starts moving the mouse after a trial's stimulus is presented and successful trial completion. Participants pressed the spacebar to confirm sucessful completion of the task. Mistrials is the number of times participants erroneously pressed the spacebar. Participants only advanced to the next trial after a successful trial completion; hence, all trials were eventually error-free.

Participants did all 3 tool conditions, with order of appearance balanced. For each tool, participants first completed a series of selection tasks, followed by layering tasks. Each series consisted of 2 trials (*low* and *high* complexity) repeated 6 times, each using a different 2D scene. Trial presentation within a session was randomized. Each one of the *high* and *low* complexity scene sets was the same across tools and users. This choice is motivated by our desire to see how the three tools perform with the same scenes. While this could have led to learning effects we did not see this in any of our analyses. Prior to using a tool for the first time, participants were presented with a brief tutorial. Then, we explained the specific task at hand. There were 4 practice trials before each task session. We told participants to do trials as quickly and accurately as possible. Participants filled a questionnaire at the end of the experiment. In summary, the study had a total of 1296 trials.

8. RESULTS

For each task, and unless specified otherwise, we conducted a 3 (tool) × 2 (complexity) × 6 (repetition) repeated measures analysis of variance (RM-ANOVA) on both the logarithmically transformed Trial Time and on the raw Mistrials data. Using the logarithm transform corrects for the skewing present in task time data and removes the influence of data outliers. We are interested to see how tools perform time-wise at different complexity levels. Because of this, our planned comparisons examine tool × complexity interactions using Bonferroni-corrected pairwise comparisons. Presentation order had no effects on either the *Trial Times* or *Mistrials* for both task conditions.

8.1 Selection Task: Trial Time

We observed a main effect for tool ($F_{2,24}$ = 7.103, p<.004). Pairwaise comparisons revealed that Palette and Splatter were significantly faster than Tumbler (p<.004 and p<.009 respectively). There was no tool × complexity interaction ($F_{2,24}$ = .357, p<.703). Our planned comparisons reveal that Palette was faster than Tumbler (p<.034) and that Splatter was faster than Tumbler (p<.031) for *low* complexities. Since the visual search time should be small for the *low* condition, we infer that the activation time of the Tumbler (an observed time between 1 and 5 seconds) accounted for most these differences. For *high* complexities only Tumbler and Palette times were significantly different, with Palette being faster (p<.015). Figure 6 illustrates the above.

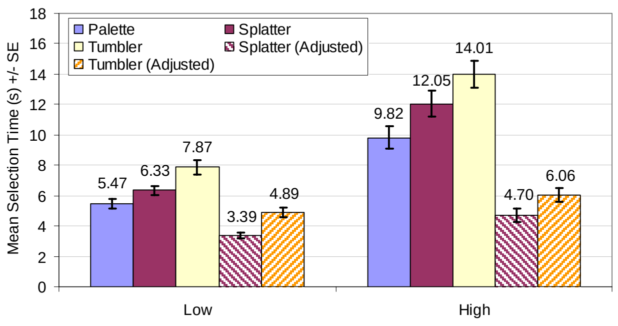

Figure 6: Complexity vs. Selection Trial Time by Tool.

We observed a main effect for complexity ($F_{1,12}$ = 338.931, p<.0001), which can be explained by the need for visual search. There was a main effect for repetitions ($F_{5,60}$ = 6.934, p<.0001) that is consistent with participants becoming familiar with the tasks as they progressed.

8.2 Selection Task: Adjusted Trial Time

Since participants did not know *a-priori* where the target was located in a scene for a selection, they likely first engaged in visual search to find the target, followed by the selection itself. We can estimate the moment when the visual search for the target in the drawing ends. We say this moment takes place the last time the Tumbler or Splatter was invoked before a successful trial (we cannot determine when the visual search ends with Palette). This definition is reasonable if we assume that users know where the object of interest is located in a scene (e.g. where there an occlusion group that needs manipulation). With this definition, these adjusted trial times still include the time it takes the user to fully open, or activate, a particular tool. Because of our experimental design, any conclusions inferred from this adjusted trial times should be taken *only* as an indication of what one may observe if one does an experiment that truly isolates visual search from the selection task.

A similar RM-ANOVA was carried out for these adjusted, logged trial times. For the these times, there was a main effect for tool ($F_{2,24}$ = 15.897, p<.0001). Pairwise comparisons showed that Splatter was faster than both Palette (p<.0001) and Tumbler (p<.031). This suggests that Splatter can be advantageous for selection tasks when users know where their target is in a drawing. There was also a tool × complexity interaction ($F_{2,24}$ = 6.647, p<.005). Our planned comparisons show that for *low* complexities Splatter was faster than Palette (p<.004) and Tumbler (p<.004). For *high* complexities Splatter was faster than Palette (p<.0001) and Tumbler was faster than Palette (p<.007). In this latter case, the Tumbler's activation time was observed to be shorter when compared with the visual search the Palette seems to impose. Figure 6 illustrates these effects.

There was a main effect for complexity ($F_{1,12}$ = 74.115, p<.0001). As the complexity of a scene increased, tasks became slower. There was a main effect for repetitions ($F_{5,60}$ = 6.409, p<.0001): users were faster with practice.

8.3 Selection Task: Mistrials

Our analysis revealed a main effect of complexity ($F_{1,12}$ = 14.083, p<0.003). Participants made more mistakes as the complexity in a scene increased. There was also a main effect of repetitions ($F_{5,60}$ = 2.487, p<0.041). The average number of *Mistrials* decreased as trials progressed. We observed no other significant effects for *Mistrial* data.

8.4 Layering Task: Trial Time

For this task, we disregard the influence of visual search because: a) participants knew *a-priori* that both the hidden and target objects were in the same occlusion group, and b) the group was clearly identifiable in the scene because of its unique bull's-eye appearance (Figure 7). Our analysis revealed a main effect for *Tool* ($F_{2,24}$ = 5.952, p<.008). Pairwise comparisons indicated that Splatter and Palette were faster than Tumbler (p<.034 and p<.007 respectively). There was also a significant tool × complexity interaction ($F_{2,24}$ = 3.546, p<.045). Our planned comparisons showed that for *low* complexities, Palette was faster than Tumbler (p<.002) and Splatter was faster than Tumbler (p<.015). Figure 7 illustrates these effects.

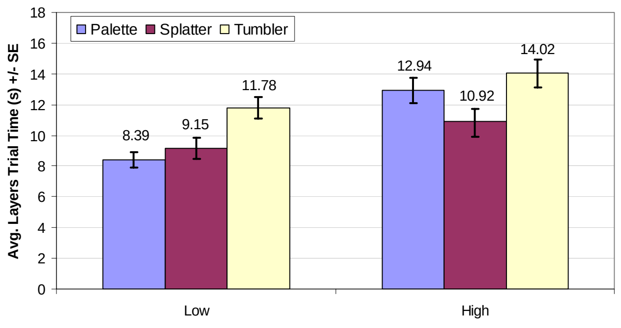

Figure 7: Layering Time vs. Complexity by Tool.

There was a main effect of complexity ($F_{1,12}$ = 33.298, p<.0001) – i.e., participants were slower for *high* complexities. Finally, there was a main effect for repetitions ($F_{5,60}$ = 1.514, p<.0001): participants were quicker with practice.

It was remarkable to see how Splatter, without strong z-value information, was at par with Palette for this layering task. We believe this was because: a) it was easier to identify both the hidden object and the targets and b) objects' proxies are usually larger or closer in Splatter than in Palette – i.e., dragging proxies one over another is faster. All times were similar for *high* complexites. These benefits go away under the presence of more distractors.

8.5 Layering Task: Mistrials

Our analysis revealed no main effects or interactions for Mistrial data.

9. USABILITY STUDY 2

To obtain usability information on a more ecologically valid task, we presented 10 participants from the first study with an optional set of jigsaw puzzle tasks. The jigsaw task consisted of presenting a target drawing on D2. Then participants were asked to re-create that drawing from a scrambled version of it presented on D1, using direct manipulation and one of three tools (Figure 8).

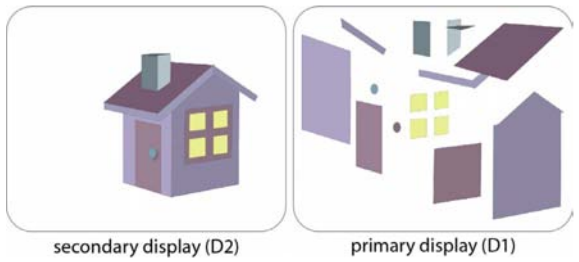

Figure 8: Jigsaw Study set-up.

For these tasks, users had the chance to use Palette, Splatter and Tumbler for two different puzzles, but not at the same time. The tools' order of presentation was counterbalanced across users. There was no measure of success, and users advanced to the next task when they decided a puzzle was solved. A short questionnaire was used to gather user opinions at the end of this study.

Although some participants had mixed feelings about the utility of Tumbler and Splatter after our first study, this real world task made them realize the full potential of these tools. It was interesting to see how users reacted to the tools, especially Splatter, after the jigsaw task. A participant's comments best summarizes these reactions: *"Based on the first set of tests, I would not have chosen the Splatter. But in the real task (the jigsaw), I worked out an efficient and sensible way of using it that felt really good. Basically, once I had the set of objects for a single group in the right basic location, I would splat them and build up the object group again, inside the splattered view, while thinking about their ordering constraints. It was very satisfying and worked well. The palette was much easier to use once the shapes were meaningful, but I still had to think about absolute ordering and exactly where something should be placed."*

There was general consensus among these participants that the jigsaw tasks gave them a better sense of how the tools would perform in the field.

10. DISCUSSION

10.1 Palette, Improved

We were reminded by our results and observations that besides being simple, the Palette is also very familiar to users. Computer users have been exposed to list selectors for many years (e.g., menus, file managers), thus adapting and developing efficient searching, selecting and manipulation strategies. Participants liked how Palette provided z-value information at-a-glance. However, they also recognized problems with this tool. Participants were concerned with the screen space the tool took as the number of objects in a scene grew. In our study, the Palette fit all the objects in the scene – up to 64. We did not study a Palette that would require a scrollbar to make objects within accesible. We believe that such a condition could negatively impact the tool's overall performance. As expected, participants consistently reported they had difficulties distinguishing objects of similar shapes and colors, but different sizes. Participants usually compensated for this problem by using Palette in concert with the scene itself – i.e., looking at how their actions with Palette affected the scene and vice-versa.

10.1.1 Next Iteration
Palette is a global tool. Unlike Tumbler or Splatter, Palette offers z-value information about objects that do not occlude. Having a *contextual* Palette that only shows the objects in an occlusion group seems like a sensible alternative. By doing this, we can reduce the number of objects referenced and the space occupied by the tool, which would result in faster selection and layering tasks times. Such a *contextual* Palette has the potential to revert its behavior to one of a global Palette, giving users the freedom to decide the scope of their interactions with the tool.

10.2 Tumbler, Improved
Tumbler looked like a promising tool, but a number of design issues conspired against its performance. Chief among these was the tool's activation time. We observed that participants waited until the tool was laid out just right – i.e., making sure layers were properly separated and oriented. This problem was aggravated with occlusion groups that generated a Tumbler with a stack too large to fit on the display. Participants also had some difficulty selecting an object. A layer's translucency gave the misleading affordance of objects underneath being selectable.

Participants also gave positive feedback about Tumbler. In particular, they praised how well it revealed the layered structure of an occlusion group. For some participants, the tool felt "useful" as well as "easy and intuitive" on the layering tasks. Participants especially appreciated the local nature of the tool, which did not require them to divert their time in creating or finding an occlusion group in Palette. They also commented that despite the axonometric distortion, objects' relative sizes sufficed in helping identifying an object of interest in real tasks.

As the number of layers in Tumbler increases above 10, its usability became hindered, due to the space the tool required. However, Tumbler is designed to be a local, contextual tool, where usability and anecdotal information tells us the number of occluding objects remains within bounds.

10.2.1 Next Iteration
Tumbler*'s* performance can increase significantly if its activation time is reduced. We can achieve this by incorporating a default spring-loaded activation, where the Tumbler decides on a proper initial orientation and separation of its layers. Also, Tumbler can keep its size within bounds by using an appropriate scaling factor based on the area of the occluding group. Furthermore, Tumbler's depth can be decreased by manually collapsing or dropping layers from its stack. Participants rarely used Tumbler to adjust an object's position during the jigsaw task. By making this operation modal we can further reduce erroneous selections.

10.3 Splatter, Improved
Splatter was an exercise in exploring the design space of tools to access and manipulate occluded content in 2D drawings. The tool demonstrated unexpected and positive performance both during our studies and later in the users' feedback. Participants praised Splatter's effectiveness, ease of use, speed and how it helped them during selection tasks. Keeping a compact, non-occluding set of proxies that maintained their relative size was well received and helped users quickly identify their target. Also, when participants knew the whereabouts of an object, Splatter presented just the objects they needed to see. Like Tumbler's case, participants expressed that the proxies' relative sizes facilitated proper identification of the object of interest.

Still, our observations revealed that for some participants, Splatter's lack of at-a-glance z-value information made layering challenging. Splatter's performance is hindered when the number of splattered objects increases above a threshold. However, Splatter is designed to be a local, contextual tool that operates on occlusion groups. Usability and anecdotal data tells us the number of occluding objects within such groups remains within acceptable bounds.

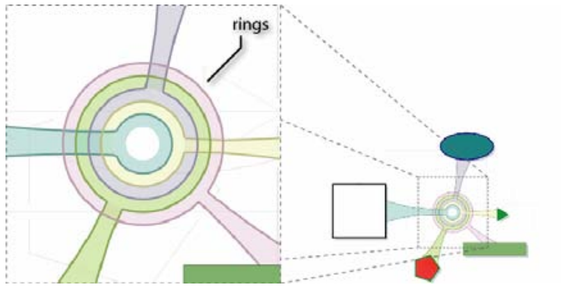

Figure 9: Splatter's Depth Well in detail.

10.3.1 Next Iteration
Splatter's handling of z-value information fueled an improvement in its design. This improvement comes in the form of a *Depth Well* (Figure 9), a novel compact visualization that capitalizes on Splatter*'s* beams to provide glanceable z-value information. The well consists of a series of concentric rings that are ordered according to the splattered object's z-value – outer rings represent objects on top of objects connected to inner rings. Also, each ring is connected to and shares the color of its related beam. Informal user feedback shows positive reaction to the well's design.

11. CONCLUSIONS AND FUTURE WORK
In this paper, we articulate a design space for techniques that help users access and manipulate occluded 2D objects. We use this space to frame an existing solution (i.e., *Palette*) and to propose two novel tools: *Tumbler* and *Splatter*. Our studies indicate that Tumbler and Splatter have potential. Results show how Splatter can outperform Palette if users know the occlusion group they want to operate on. By design, Palette, Tumbler and Splatter can co-exist and complement each other well. Study participants agreed almost unanimously that they could see themselves using a particular tool for a particular scenario. Offering users the freedom to choose the best tool for the job was considered valuable to them as well as a significant contribution from our work.

Future work includes revising, improving, and further exploring the proposed design space as well as implementing and evaluating next iterations for Tumbler, Splatter, and Palette. We would like to add support for group management and layer hierarchies into these new iterations.

12. ACKNOWLEDGMENTS

We would like to thank Dan Robbins and Georg Petschnigg for their graphics and UI design insights, Dan Olsen and Ed Cutrell for help with experimental design and analysis, and Amy Karlson, Tara Matthews and Johnny Lee for support and encouragement.

13. REFERENCES

[1] Apple. Exposé, http://www.apple.com/macosx/features/expose/.

[2] Baudisch, P., et al., Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-Operated Systems. (2003) Interact, p. 57-64.

[3] Baudisch, P., et al., Multiblending: Displaying Overlapping Windows Simultaneously without the Drawbacks of Alpha Blending (2004) CHI, p. 367-374.

[4] Beaudouin-Lafon, M., Novel Interaction Techniques for Overlapping Windows (2001) UIST, p. 153-154.

[5] Bier, E., et al., Toolglass and Magic Lenses: The See-through Interface (1993) SIGGRAPH, p. 73-80.

[6] Carpendale, M.S.T., et al. Extending Distortion Viewing from 2d to 3d (1997). *IEEE CG&A*, *17* (4).p. 42-51.

[7] Dragicevic, P., Combining Crossing-Based and Paper-Based Interaction Paradigms for Dragging and Dropping between Overlapping Windows (2004) UIST, p. 193-196.

[8] Inselberg, A., et al., Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry (1990) VIS, p. 361-378.

[9] Ishak, E.W., et al., Interacting with Hidden Content Using Content-Aware Free-Space Transparency (2004) UIST, p. 189-192.

[10] McGuffin, M.J., et al., Using Deformations for Browsing Volumetric Data (2003) VIS, p. 401 - 408.

[11] Ramos, G., et al., Fluid Interaction Techniques for the Control and Annotation of Digital Video (2003) UIST, p. 105-114.

[12] Robertson, G., et al. The Large-Display User Experience (2005). *IEEE CG&A*, *25* (4).p. 44.

[13] Shen, C., et al., Cor2ds (2005) CHI Ext. Abs., p. 1781-1784.

[14] Sonnet, H., et al., Integrating Expanding Annotations with a 3D Explosion Probe (2004) AVI, p. 63-70.

[15] Stange, K. Dr. Miro (2000), http://kenstange.com/drmiro/.

[16] Ware, C., et al. Motion to Support Rapid Interactive Queries on Node--Link Diagrams (2004). *ACM Trans. Appl. Percept.*, *1* (1).p. 3-18.