Mixed-Initiative Approaches to Global Editing in Slideware

Darren Edge[1], **Sumit Gulwani**[2], **Natasa Milic-Frayling**[3], **Mohammad Raza**[3], **Reza Adhitya Saputra**[1,4], **Chao Wang**[1], **Koji Yatani**[1,5]

[1] Microsoft Research	[2] Microsoft Research	[3] Microsoft Research	[4] University of Waterloo	[5] University of Tokyo
Beijing, China	Redmond, USA	Cambridge, UK	Waterloo, Canada	Tokyo, Japan

{daedge, sumitg, natasamf, a-moraza, chaowa}@microsoft.com, radhitya@uwaterloo.ca, koji@iis-lab.org

ABSTRACT

Good alignment and repetition of objects across presentation slides can facilitate visual processing and contribute to audience understanding. However, creating and maintaining such consistency during slide design is difficult. In order to solve this problem, we present two complementary tools: (1) StyleSnap, which increases the alignment and repetition of objects by adaptively clustering object edge positions and allowing parallel editing of all objects snapped to the same spatial extent; and (2) FlashFormat, which infers the least-general generalization of editing examples and applies it throughout the selected range. In user studies of repetitive styling task performance, StyleSnap and FlashFormat were 4-5 times and 2-3 times faster than conventional editing, respectively. Both use a mixed-initiative approach to improve the consistency of slide decks and generalize to any situations involving editing across disjoint visual spaces.

Author Keywords

Presentations; visual consistency; layout editing; snapping; programming by example; least-general generalization.

ACM Classification Keywords

H.5.2. Information interfaces and presentation: User Interfaces.

INTRODUCTION

Designing, delivering, and watching slide presentations are common aspects of professional life. As a modular authoring medium, slides constrain the problem of visual communication and afford flexibility in restructuring and reuse. However, this same modularity can result in weak connections between slides that are also visually inconsistent.

The theory of *processing fluency* proposes that aesthetic pleasure is based on perceptual processing, with high fluency associated with positive evaluations such as agreement [33]. For slide design, audience negativity toward visual disorder (e.g., clutter, inconsistency, misalignment) has been argued to result directly from reduced processing fluency [1].

CHI 2015, April 18 - 23 2015, Seoul, Republic of Korea. Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3145-6/15/04…$15.00
http://dx.doi.org/10.1145/2702123.2702551

Nancy Duarte, author of "Slide:ology" [10], recommends the reuse of slide layouts to help the audience anticipate where content will appear next (i.e., to increase processing fluency through repetition). Garr Reynolds, author of "Presentation Zen" [34], also recommends repetition, but of "certain design elements" rather than whole slide layouts. The use of slide templates, copied slide elements, and temporary grids can all help to avoid inconsistency problems, whether within slides (nonaligned elements appearing randomly placed), between slides (misaligned elements "jumping" on transition), or across the deck (related elements lacking consistent styling).

However, creating and maintaining visual consistency across slides is difficult when the desired layouts and styles are not known in advance. During the design of slide visuals, making systematic changes one element at a time is both repetitive and tedious. Ad-hoc changes also make it difficult to refactor content using slide templates. Lack of support for the consistent redesign of elements repeated across slides is thus a major contributing factor to low processing fluency.

To address both the consumption problem of processing fluency and the authoring problem of repetitive styling, we present two complementary tools: (1) *StyleSnap*, for the automatic alignment of object edges within and across slides; and (2) *FlashFormat*, for the systematic restyling of related objects. Both are *mixed-initiative systems* [19] in which users collaborate with intelligent services to achieve their goals.

The intelligence of StyleSnap is through progressive hierarchical clustering of each of the four object edge positions (offsets from the left and top slide edges), in ways that increase alignment both within and across slides without introducing object overlaps or image distortions. Following an automated snapping process, the user can independently unsnap, merge, and style the resulting groups of objects that have been snapped to the same slide extent (i.e., all four corresponding edges share the same position values).

Conversely, the intelligence of FlashFormat comes after the user has directly supplied some examples of the repetitive edit they wish to apply more generally. It infers the *least-general generalization* [32] of the example edits through the attribute values shared among all edited objects and the transformation performed on the edited objects. The next application of FlashFormat always extends the current edit to the smallest set of objects that can be generalized to by allowing the unshared attributes to vary freely.

Since StyleSnap increases the number of shared attribute values throughout a slide deck (via both snapped object positions and propagated style changes), prior use of StyleSnap means that fewer example edits are required to cover all of the variance in the set of objects to be edited with FlashFormat. The two tools are thus complementary, although each has substantial value in standalone use.

In this paper, we first present a literature review on snapping and formatting, as well as the tools available in commonly used graphical systems. We then present an examination of a PowerPoint slide-deck corpus, showing how the use of such templates in practice restricts their ability to create and maintain consistency. In the following sections, we describe the design and implementation of two tools—StyleSnap and FlashFormat—that address the limitations of conventional templates. We also present a user study of each, demonstrating their superiority over existing approaches in task performance time and user preferences. We then report the lessons learned from both systems and discuss improvements for future mixed-initiative editing systems. These are relevant to the category of slideware augmented in this paper, as well as to any visual software that requires consistent styling across multiple pages (e.g., word processor documents), grids (e.g., spreadsheets), frames (e.g., editing of video overlays), or repeated visual spaces of any kind.

Overall, this work offers the following contributions:

1. Two approaches to selecting objects for systematic editing, based on: (a) similarity of objects' spatial extents as determined by clustering of their edge positions (StyleSnap); and (b) similarity of objects' attribute values as determined by the least-general generalization of example edits performed on other objects (FlashFormat).
2. Two approaches to editing objects simultaneously in order to reach a state of consistency, based on (a): snapping misaligned object edges into a reduced number of aligned positions, without introducing artifacts (StyleSnap); and (b) applying the least-general generalization of example edits to all objects selected for editing (FlashFormat).

RELATED WORK

Microsoft PowerPoint remains the most popular application in the category of "slideware". It has a wealth of existing presentations available online, rich tools for the analysis of its XML-based documents, and an add-in framework for augmentation of its feature set. Therefore, in the development and evaluation of our tools, we take PowerPoint as representative of slideware as a whole.

Alignment of Document Objects

Snapping is a commonly used technique to guide objects into a state of alignment [5]. For example, PowerPoint 2013 provides visual line-guides and snapping behavior to support object alignment within a slide, as well as the traditional snapping of object edges to an underlying grid whose resolution can be customized. Object snapping can be extended in several ways, such as adaptively changing the snapping behavior based on user input [26] or adaptively inserting motor space to support differentiation of snap targets [4]. Other approaches have addressed mouse-free snapping on surfaces and tabletops by using control-display gain [13], multi-touch [14], pen input [15], and the non-dominant hand [40]. None of them focus on alignment across disjoint visual spaces of any kind.

Layout of Document Objects

One approach to achieving consistent, well-designed layouts is to allow the user to select from a range of predesigned "templates" and enter content accordingly. Instead of making layout changes on individual slides (which can create inconsistencies), the user makes changes on the template itself, automatically updating slides that adopt the template.

In PowerPoint 2013, templates are provided in a special view called the Slide Master. This view defines slide layouts and styles through *placeholders* which are also exposed when the user selects a layout for a new slide from a menu or when the user updates a slide to an alternative Slide Master layout. However, there are several usability problems with this approach. First, templates need to be selected in advance of slide creation rather than formed through exploration on slides. Second, directly editing a slide object that is linked to a placeholder removes the ability to restyle that object through the Slide Master, unless the slide template is manually re-applied. When inspecting either an object or a slide, it is not possible to determine whether the links to the placeholder are intact. That makes it difficult to anticipate the scope of changes that are possible through the slide templates. All of these specific problems belong to a more general category of risks that are known to reduce users' willingness to invest attention in abstraction use [6].

In the research domain, document analysis has been used to suggest layouts that satisfy criteria given by the user [24], such as logical structures of information to be visualized in presentation slides [38], or to support version management of multiple slide decks [9]. An alternative approach is to specify the structure of the desired document directly and allow the system to provide "styling as a service", as in the HyperSlides system for presentation prototyping [11].

One approach to automating layout is to consider it as a visual constraint satisfaction problem. Constraints express high-level relationships between objects (e.g., text referencing a picture) or geometric structures (e.g., the sizes of all textboxes are the same). These constraints can be described as rules [7, 16, 39]. A major challenge of such rule-based systems is to anticipate all required rules. Another approach is to consider layout automation as a problem of optimizing energy functions. This has been explored in the context of adaptive grids [20], focusing on the goodness of template fit and micro-typography aesthetics. The concept of "conditional shapes and groups" can help to generate more flexible constraint systems dynamically [36], as can combinations of constraint- and force-based approaches [2].

While such automated layout systems are good for content that would not otherwise be "designed" or is yet to be designed, our tools specialize in the restyling of existing presentation content with a high degree of user control.

Formatting of Document Objects

The Slide Master is a form of indirect editing in PowerPoint and similar slideware. Another is the Format Painter, which allows all non-spatial attributes of a source object to be copied and "painted" onto destination objects, thus supporting reuse of object formatting in the slide view.

Macros, batch processing, history brushes, and graphical search and replace [22] are further ways in which users can capture and reuse action sequences. EAGER [8] is an early work using programming-by-example principles [29] to support efficient text data entry but not visual style changes. Abstract object selection and restyling are also possible with interactive machine learning [12], by inferring the user's desired scope based on patterns of selection and deselection (e.g., for file selection [35] and friend selection [3]).

For document editing, the LAPIS text editor [27] supports intelligent group selection and simultaneous reformatting of strings. An extension supports intelligent find-and-replace operations in text documents, grouping different string selection candidates by literal and semantic similarities [28].

CORPUS ANALYSIS OF TEMPLATE USAGE

We wanted to understand the extent to which the apparent usability problems of Slide Master templates were evident in examples of presentations downloaded from the Web. We built a corpus of over 8000 presentations using Bing web searches specifying the "ext:pptx" filter. The resulting slide decks were drawn from many fields including business, government, science, technology, and education. We used the Open XML SDK 2.5 [30] to parse these presentations and extract statistics on slides and objects. Table 1 shows the results from the 7663 successfully processed files.

We found that 88% of slides (24.7 out of 28.2) on average contained placeholder objects created by the Slide Master. We also found that the proportion of objects per slide that could actually be restyled through the Slide Master (without first resetting the slide to restore broken links, which loses any custom layout and styling of placeholders) was only 21% (1.1 out of 5.3) on average. Thus, while the vast majority of slides (88%) have the potential to be updated via the Slide Master, due to user editing behavior the vast majority of objects (79%) still require manual editing.

We conducted a second analysis to understand the degree to which objects shared the same or similar *spatial extent* on slides, reflected as the same or similar edge offsets (distances from the left and top slide edges). For each deck, we first created a mapping from extents to objects, progressively varying the matching tolerance from 0 to 100 points in 10 points increments (28pt=1cm). In each iteration, we grouped objects across slides whose edges were all located within a matching region of an existing extent (e.g., for the tolerance of 0, we only grouped objects with identical extents).

Slide Statistics	Mean (*SD*)
# of slides	28.2 (20.7)
# of slides with placeholders unmodified	19.0 (17.5)
# of slides with placeholders modified	5.7 (9.0)
# of slides without placeholders	3.5 (7.8)
Shape Statistics	**Mean (*SD*)**
# of objects per slide	5.3 (6.1)
# of objects created by user	3.9 (6.2)
# of objects from unmodified placeholders	1.1 (0.9)
# of objects from modified placeholders	0.3 (0.4)

Table 1. Statistics on the number of slides and objects in 7,663 PowerPoint files we collected from the Internet.

This analysis revealed a strong skew resulting from many small, often single-element groups. With exact matching (a tolerance of zero points), the average number of position groups was 35 and objects in the largest position group occurred on 82% of slides. All groups in the upper quartile (top 9 groups of 35) had objects occurring on at least 5% of slides. Since these cannot all be placeholder objects (which only contribute 1.4 objects per slide), they are likely to come from copy-and-paste actions applied to ensure size, position, and style consistency for individual objects (and slides). When the tolerance was 60 points (about 2cm – a "near match" on all four object edges), the number of overall position groups halved (from 35 to 18). Objects in the largest position group occurred on 93% of slides and those in the upper quartile (top 5 groups of 18) had objects occurring on at least 11% of slides.

In conclusion, there is ample opportunity to increase the *positional consistency* of objects by mapping "near match" objects into the exact same extent. If such a system could be developed, it could also increase the *style consistency* of position groups by propagating style changes to all members of the associated group. Such a system would have a greater restyling power than the Slide Master while also offering *implicit object templates,* abstracted directly from slides, rather than *explicit slide templates* designed indirectly in a separate view. Many of the problems of template-based layouts could thus be avoided.

STYLESNAP

Following on from the previous corpus analysis, we designed a mixed-initiative tool called *StyleSnap* that can be applied whenever a slide deck has evolved into a state of misalignment or inconsistency. Our first goal was to develop a method to align objects across slides without introducing new problems, such as object overlap and image distortion. Our second goal was to develop a user interface that would allow a user to invoke StyleSnap, view the resulting groups of objects snapped to the same extent, then undo, merge, or modify these groups accordingly. Our implementation of these concepts was through an add-in for PowerPoint 2013. We now describe the high-level design of the StyleSnap user interface and details of the underlying snapping algorithm.

StyleSnap Interface

Figure 1 shows the result of pressing the "StyleSnap" button in the PowerPoint ribbon menu. A side pane appears showing the resulting position groups—groups of objects across slides that have been mapped to the same position and size as a result of clustering and snapping edge values for the four edge types. Each group is listed showing its color (matching the color of the highlight boxes added to the corresponding objects), the number of objects in the group, a "Style Painter" icon for manually adding objects to the group and merging groups with one another, and a checkbox indicating whether the group is currently in its "Snap" state. Only non-singleton object groups are snapped by default but the user can toggle snapping for both individual groups and for all objects. Snapped objects are designated with a solid border in their new extent; unsnapped objects with a dashed border in their original extent. If all objects were already in the same extent and unaffected by snapping, the Snap checkbox is disabled.

Clicking on any slide object or position group highlights the object, its position group in the side pane, and all other objects in the group. For efficient visual browsing of this position group, the system also gathers all slides containing highlighted objects and places them in a temporary "Selected" section at the start of the slide list, with other slides organized in an "Unselected" section (existing sections are recreated after StyleSnap tool use). The color of the highlight box of the selected group fully saturates for clear differentiation from the colors of other object groups. This visual feedback allows the user to confirm quickly whether the automatic snapping of the position group is desirable. When using the StyleSnap tool, any location, size, font, or other format changes to an object are propagated across all other objects in its group and made visible in real-time on all slides in the "Selected" section of the slide list.

The Style Painter is similar to the Format Painter but extends to position and size attributes. Activating the Style Painter for a particular group by clicking the corresponding icon means that the next selected object or position group will automatically be merged with the active group.

Clicking the "Apply" button saves the changes to the deck and reverts to standard editing. Clicking "Discard" undoes the changes and reverts the deck to its pre-StyleSnap state.

Snapping through Hierarchical Clustering

We wanted to enable "snapping" of objects into (a) fewer extents with (b) better alignment across slides. Each of these goals suggests a different approach. In order to reduce object extents, we could apply hierarchical clustering [18, 37] directly to object extents (since it does not require a prior choice for the number of clusters, unlike, e.g., K-means). However, this will not result in good cross-slide alignment if the extents of objects within slides are misaligned. There is also a sparsity problem—objects may all be sufficiently different and no two objects should be mapped to the same extent, even though any individual object edge may be close to the corresponding edges of many other objects.

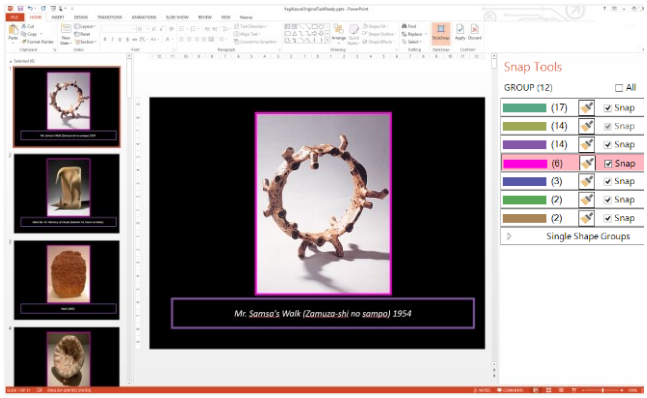


Figure 1. StyleSnap interface. The right pane shows all the groups whose objects are moved to the same position as a result of snapping. The user can manually undo and merge position groups, as well as simultaneously style all objects in the same group with real-time feedback in the slide list on the left.

This led us to a solution based on hierarchical clustering of individual object edges rather than object extents. We independently perform hierarchical clustering of all the left, top, right, and bottom edge positions of the objects within a presentation and then apply the results back to the objects. The result of such clustering for each of the four edge types is a hierarchy of depth N where N is the number of distinct edge positions. At level 1 there are N clusters of 1 edge position. At level N there is 1 cluster of N edges, all snapped to the modal edge position. From level 1 to level N, the algorithm merges the closest pair of clusters and snaps all edges in the new cluster to the new modal edge position.

We can determine the optimal clustering level for each edge type (left, top, right, and bottom) by an energy function that aims to balance similarity within and between clusters (i.e., to make close edge values the same while keeping distant edges separate). However, the naïve updating of object edge positions based on the optimum clustering of each edge type could easily cause problems in visual appearance:

1. *Object inversion*, since independent edge clustering does not respect relative positions of opposing object edges;
2. *Object overlap*, since initially separated objects can be moved into a state where their content regions overlap;
3. *Image distortion*, since images are especially sensitive to aspect ratio changes.

Given that we can identify and correct these problems only after position updates, our divide-and-conquer approach of clustering different edges independently is most suited to the agglomerative, bottom-up method of hierarchical clustering supported by the SLINK algorithm [37]. Overall, our snapping approach is summarized in Algorithm 1.

Systematic Performance Evaluation of Snapping

Snapping completes in several seconds, even for 500 objects (the 97th percentile in our PowerPoint corpus). In order to evaluate the degree to which automated snapping matches human judgments, we recruited two professional software engineers to test its accuracy and reliability across a range of

typical slide decks: 80 presentations from our PowerPoint corpus, with varying numbers of slides (15–36; the lower and upper quartiles) as well as varying numbers of objects per slide (2.5–6.0 with mean 4.7).

Table 2 shows how the number of singleton position groups (a mean of 53.6) forms a long-tail, as expected from the earlier corpus analysis. Problems resulting from snapping are well controlled in both group types, limited to overly shrinking objects, creating inconsistencies among sets of slide objects that were previously consistent (e.g., in size, spacing, or alignment), and breaking spatial relationships between objects (e.g., by moving arrows within diagrams). Shrinkage can be dealt with by adding simple rules, but to address problems that arise from object relationships would require more complex rules or manual object grouping. Overall, on average only 4.5 position group modifications were required to reach a satisfactory state of alignment.

EVALUATION OF STYLESNAP

We conducted two user studies to evaluate the performance of StyleSnap against two alternative approaches to cross-slide alignment and styling: *Repeat Editing* and *Slide Master*. For the study tasks we selected a slide deck from our internet corpus with an average of 2 objects on each of 31 slides—one from a Slide Master template, one added manually—with a balance of title-and-bullets and image-and-caption slides.

Task: Cross-Slide Alignment of Misaligned Objects

There were 14 slides containing one picture and one textbox used as a caption, with no perfect alignment of any pair of objects. The task was to align all four object edges of groups of images with similar aspect ratios and make the caption format the same across all 14 slides. For consistency, we defined target object groups for the 14 images based on three aspect ratios: 9 portrait images, 4 landscape, and 1 panoramic. We also set the target text format to be 20pt Red Italic Arial.

Expert Performance Prediction

In order to predict the expert performance of each technique, we adopted an approach akin to Keystroke Level Modelling (KLM) by constructing a task model for each system and quantifying its time parameters through a user study. Since these models do not account for switching costs, they represent predicted lower bounds on task completion time.

***Repeat Editing*:** [*Start time* T_{re}] Make a duplicate of Slide 13 and use it as a layout guide to align content from Slide 14. Delete unwanted objects and slides. [*End time* T_{re}]

Reference task time = T_{re} *× 13 slides*

***Slide Master*:** [*Start time* T_{sm1}] Open the Slide Master view and create a layout with content placeholders in the desired positions and aspect ratios. Close the Slide Master. [*End time* T_{sm1}] [*Start time* T_{sm2}] Go to Slide 13 and update it to the new custom layout. Delete unwanted objects [*End time* T_{sm2}]

Reference task time = T_{sm1} *× 3 slide layouts (one per aspect ratio) +* T_{sm2} *× 14 slides (apply new layout to each)*

Algorithm 1. Global snapping of object edge positions

1. **Cluster object edge positions in preparation for snapping**
   For each edge type *T* (left, top, right, bottom):
   a. Cluster edge positions using the SLINK algorithm
   b. Calculate the optimal clustering level L_T to minimize the following distance-based energy function:
   $$\alpha \sum_{c \in G}(C_g - C_{global}) + (1-\alpha) \sum_{g \in G} \sum_{s \in g}(l_s - C_g)$$
   (C_g, C_{global}, and l_s represent the centroid edge value of each cluster, the one of all objects, and the edge value of one object, respectively. α is the coefficient to determine weights for between-cluster and within-cluster similarities, set to 0.5)
2. **Apply snapping progressively, resolving overlaps and inversions**
   For $i = 1$ *to* N (number of adaptive steps, e.g., 20):
   a. For each edge type *T*:
      i. Apply the snapping associated with clustering at level $[(i/N) \times L_T]$ (where level 1 is no clustering) to objects that have not previously been inverted or overlapped.
   b. For each object O_i:
      i. If O_i is newly inverted or overlapped, reset to O_{i-1}.
   c. If no objects can be snapped further, move to (3).
3. **Resolve image distortion**
   For each snapped image object *I*:
   a. If its aspect ratio of *I* has changed by more than a threshold proportion (set to 0.1), shift the furthest-moved edge of *I* to return to the original aspect ratio, otherwise allow the change.

Position Groups of 2+ Shapes	Mean (*SD*)
# of position groups	10.3 (10.0)
# of position groups where snapping causes problems	0.7 (1.2)
# of merges among 2+ position groups required	0.8 (1.1)
Position Groups of 1 Shape (Singletons)	**Mean (*SD*)**
# of position groups	53.6 (33.7)
# of position groups where snapping causes problems	0.6 (2.5)
# of additions to 2+ position groups required	2.4 (2.8)

Table 2. Performance of StyleSnap auto-alignment on 80 decks.

***StyleSnap*:** [*Start time* T_{ss1}] Use the Style Painter to add the portrait image on Slide 28 to the position group of the portrait image on Slide 13. [*End time* T_{ss1}] [*Start time* T_{ss2}] Set the caption text to the target format. [*End time* T_{ss2}]

StyleSnap groups the captions into a single group and images into groups with 4, 4, 2, 1, 1, 1, and 1 objects. Reaching three position groups for these images requires a maximum of eight changes using the Style Painter (one object at a time).

Reference task time = T_{ss1} *× 8 group changes (to make 3 groups) +* T_{ss2} *(applies to all captions at once)*

Procedure

We first explained and demonstrated each task before asking participants to repeat the procedure until they had mastered it. We used two such slides for our demonstration, with the task to make the layout and style of Slide 14 the same as Slide 13. We offered detailed instructions for each approach and allowed several timed trials until we did not observe large time variances. Participants' best times were recorded and used to calculate the reference task time (as is common when seeking to understand expert performance, e.g., in [40]).

We recruited 12 participants (6 male and 6 female, the average age of 25; PA1–PA12) from a local university. All were familiar with PowerPoint and fluent in English. We counterbalanced the condition order. Cash equivalent to $15 USD in local currency was offered as compensation.

Expert Performance Results

Table 3 summarizes the completion times of sub-tasks in each of the three techniques. Entering these times into our task models predicts the average expert performance to be 353, 322, and 107 seconds with Repeated Editing, Slide Master and StyleSnap, respectively. This prediction is highly promising and indicates that StyleSnap could substantially reduce the user effort of cross-slide alignment and styling. The results also show that updating any number of target slides with StyleSnap is comparable to refactoring a single slide with Slide Master, once the StyleSnap groups are correct and the Slide Master templates are created. Since 4.5 StyleSnap group merges are required to correct the snapping of a whole deck, on average, it also means that StyleSnap is about as fast at correcting alignment throughout this particular deck and refactoring 14 slides of a particular design (69s) as Slide Master is at creating a single template and refactoring just two slides (71s).

StyleSnap Task for Expert Users: Completion Times	Mean (*SD*)
T_{re}: Refactor a target slide using Repeat Editing	25.2 (6.4)
T_{sm1}: Make a slide template using Slide Master	43.2 (9.6)
T_{sm2}: Refactor a target slide to a slide template using Slide Master	13.8 (2.1)
T_{ss1}: Find and add an object to a group using StyleSnap	11.5 (2.7)
T_{ss2}: Update all target slides using StyleSnap	15.2 (5.6)

Table 3. Sub-task completion time results in the expert performance user study.

Novice User Performance Measurement

We also conducted a supplementary study with the same task set to measure the performance of novice users. The procedure was the same except that participants were asked to modify all 14 slides in each task. This study therefore offers realistic performance observations of the three techniques that account for the effects of learning and fatigue. From the expert performance prediction results, we had two hypotheses for this study: **[HA-1]** StyleSnap would be faster for global alignment than existing tools; and **[HA-2]** the perceived workload of global alignment tasks would be smaller with StyleSnap. StyleSnap training included 5-10 minutes of free exploration on a range of decks. Following the study tasks, participants completed a NASA-TLX questionnaire [17] for each approach. We recruited another eight participants for this study (5 male and 3 female, the average age of 25; PB1–PB8) from our research institute.

Novice User Performance Results

One-way repeated-measure ANOVA revealed a significant difference among techniques ($F_{(2,14)} = 90.8$, $p < .0001$, $\eta_p^2 = .93$), with *StyleSnap* faster than the other two ($p < .0001$ for both), shown in Figure 2, supporting HA-1. Analysis of NASA-TLX responses found seven significant pairwise results as shown in Table 4, partially supporting HA-2.

While the cost of both *Repeat Editing* and the *Slide Master* scale linearly with the number of target objects, the cost of StyleSnap scales only with the number of corrections to groups that contain target objects. The fewer groups that need to be corrected, the closer the performance of StyleSnap to a constant time cost, regardless of the number of objects.

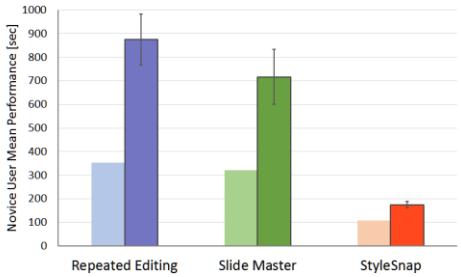


Figure 2. Predicted lower bounds on expert performance (bars without borders) and novice user mean performance time (bars with borders). Error bars represent 95% Confidence Intervals.

Qualitative Discussion

Participants in both studies unanimously preferred *StyleSnap* over *Repeat Editing* and *Slide Master* and many commented on how it would support their everyday authoring. The Slide Master was also widely criticized, e.g.: *"If we edit using Slide Master, we really don't know what the final result will look like. After we edit it we should go back to see the result… if is not satisfying we should go back again to the slide master view... If we use StyleSnap we can see the result directly"* (PA12).

The cost of automated snapping and manual modification was perceived to be relatively low. Participants described the "clustering" as *"very efficient… it really helps in saving time"* (PA4); *"really useful"* (PA8); *"a great idea… a simpler way to edit a lot of shapes"* (PA9). Participants also appreciated being able to edit all objects in a position group simultaneously: *"I can do quick edits for the whole content in the slides"* (PA5); *"You can do one 'style' and apply it everywhere fast"* (PA2).

The study also highlighted areas for improvement. Several participants expressed initial confusion about the meaning of colors and suggested the use of semantic descriptors, icons, or thumbnails to make the mapping clearer. One participant found the "shuffling" of the slide list distracting while another suggested a drag-and-drop mechanism for Style Painting. During free explorations of multiple decks, several participants expressed a desire for an algorithm that automatically identifies diagram-like collections of objects and groups them for StyleSnap alignment purposes.

StyleSnap Task for Novice Users: NASA-TLX Subjective Workload Results (in the format *median* [*lower quartile*, *upper quartile*])							
	Mental	Physical	Temporal	Performance	Effort	Frustration	Weighted overall
Repeat Editing (RE)	**25.0** [15.0,47.5]	**82.5** [70.0, 96.3]	**82.5** [73.8, 92.3]	**67.5** [62.5, 75.0]	**90.0** [73.8, 100.0]	**85.0** [78.8, 91.3]	**78.0** [73.3, 81.3]
Slide Master (SM)	**47.5** [40.0, 57.5]	**47.5** [33.8, 58.8]	**45.0** [27.5, 57.5]	**22.5** [8.8, 36.3]	**52.5** [25.0, 55.0]	**52.5** [36.3, 66.3]	**43.8** [35.8, 55.0]
StyleSnap (SS)	**30.0** [18.8, 66.3]	**22.5** [18.8, 31.3]	**17.5** [13.8, 21.3]	**15.0** [5.0, 16.3]	**27.5** [25.0, 33.8]	**20.0** [8.8, 30.0]	**25.0** [22.7, 29.3]
Friedman test: $\chi^2(2)$	0.077 ($p = .96$)	11.6 ($p < .01$)	13.6 ($p < .01$)	10.2 ($p < .01$)	13.1 ($p < .01$)	14.0 ($p < .001$)	15.5 ($p < .001$)
Pairs with $p<.05$	(none)	(none)	RE-SS	(none)	RE-SS, RE-SM	RE-SS, RE-SM	RE-SS, RE-SM

Table 4. NASA-TLX subjective workload results for novice users completing the StyleSnap task. Lower values are better.

The study also surfaced tensions in the design of StyleSnap, which is predicated on the value of object alignment and style consistency throughout a slide deck. As one participant explained, *"With StyleSnap I can easily design the layouts, especially if I want to change the same layouts, not manually edit the slides. But I think it will be difficult to change if on each slide the designs are different"* (PA8). A different approach is required to make repetitive changes to objects that do not share the same position—this is the purpose of the complementary FlashFormat tool that we present next.

FLASHFORMAT

StyleSnap supports aligning objects across slides in ways that increase processing fluency, but it is not applicable to repeated-object restyling when target objects are placed at different locations. FlashFormat offers the ability to apply global style changes with more flexible object selection.

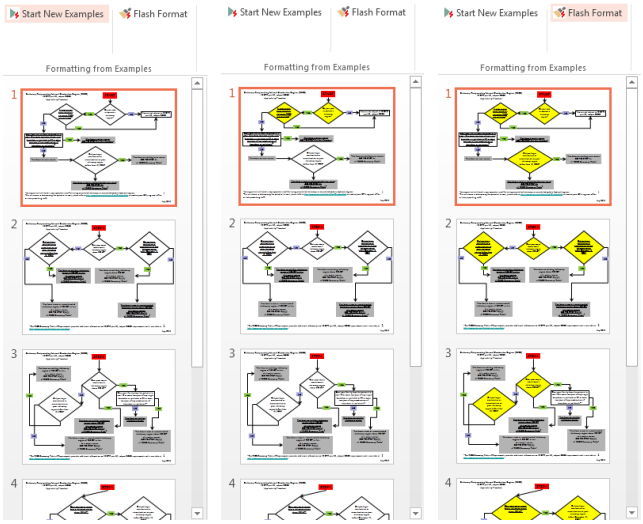
FlashFormat is a programming-by-example system [29] that allows the user to perform repetitive formatting tasks in PowerPoint. The interface of FlashFormat is shown in Figure 3. It consists of only two buttons: "Start New Examples" and "FlashFormat". The user starts by clicking "Start New Examples" and then gives some examples of the formatting changes they would like to perform. When they click on "FlashFormat", the system infers the *least-general generalization* (LGG) from the given examples and applies the changes to the rest of the document ("FlashFormat-all").

In Figure 3b the user gives two examples of changing the color of diamond shapes to yellow, for which the system infers the LGG that changes all diamond shapes to yellow (Figure 3c). The inferred generalization depends on the given examples; for instance, to color yellow any object with underlined text, the user may give examples for different shapes and colors (e.g., white diamond, white rectangle, gray rectangle) containing underlined text.

The inference performed by FlashFormat is based on the XML specifications of objects (using the OOXML file format), and hence covers all properties expressed in this specification. The system is based on a domain specific language for expressing transformations of XML structures, and a synthesis algorithm for inferring LGG programs within this language. In this work, we focus on the interaction model and usability studies of the tool. The underlying inference algorithm builds on prior work in program synthesis [32].

An interactive and incremental generalization process

The user can guide the system to the desired generalization in an interactive and incremental fashion. Since the system is conservative in the inference of the generalization, the user can give additional (dissimilar) examples to express their intended selection criteria. For example, if the user wants to color all objects with underlined text and only gives examples of diamond shapes, then the system may infer a less general hypothesis: to color only diamond shapes containing underlined text. However, the system is designed to incrementally accept examples. At that point the user can give more examples through manual editing, and then



(a) (b) (c)

Figure 3. FlashFormat interface, illustrating the coloring of all diamonds to yellow: (a) the user clicks "Start New Examples"; (b) gives two examples of diamond shapes, changing the color; (c) clicks "Flash Format" to apply the least-general generalization from the example changes.

FlashFormat again; from added examples, the system will infer a more general transformation.

The user can also backtrack if the system does not infer the intended transformation. If the inferred generalization is not applicable to any other objects, then the system applies the transformation to the closest matching objects, according to a similarity measure on the XML specification of objects. If this inference is inaccurate, then the user can undo the changes using the standard "undo" feature, and provide more examples to guide the system in the right direction.

The system also supports a restricted application of the inferred transformations, allowing the user to verify the transformation before applying it globally. After giving examples, the user can select a set of objects or slides and then click "FlashFormat" to apply the transformation only to the selected objects or slides ("FlashFormat-selected").

EVALUATION OF FLASH FORMAT

We conducted a user study to examine the use of FlashFormat. We formulated three hypotheses: **[HB-1]** Users would be able to achieve their desired global restyling through FlashFormat; **[HB-2]** with the experience of using FlashFormat, users will develop a sense of which examples to give, and **[HB-3]** across a range of tasks, using FlashFormat would be faster than standard editing tools.

Tasks: Cross-slide Shape Restyling

We prepared a slide deck downloaded from the Internet. It contained flowcharts spanning five slides, using different shapes (rectangles and diamonds). Such diagrams are commonly encountered in slide presentations and cannot be restyled through the slide master because objects invariably occupy unique slide extents. Participants were asked to

perform two changes specified by the experimenter, either with FlashFormat or manually (use of Format Painter was allowed in this condition). In order to create a balanced workload per task, we varied the number of objects to be restyled and the difficulty of giving examples. The participants were asked to make two systematic style changes in each trial but were not allowed to change any other visual attribute or text content of the objects. After this controlled task, participants were given another slide deck and asked to perform global restyling as they liked. They were encouraged to use FlashFormat to make changes and were given five minutes for this part of the study.

The interface used in the study did not include visual feedback before clicking the FlashFormat button. Our intention was to study how well participants could understand the behavior of LGG without being influenced by the feedback design and to test its effectiveness in a most basic form (any additional improvements on feedback would generally favor the performance of FlashFormat).

Procedure and Participants

We first explained FlashFormat with two examples of slide decks and asked participants to perform global restyling tasks. The slide decks included easy and difficult cases for FlashFormat. This pre-task session was intended to make participants knowledgeable about the system and able to perform restyling without help from the experimenters. We provided explanations that choosing *more, and more diverse examples* would lead to better results (referred to as "the golden rule"), but we did not force participants to do so. During each trial, we measured the performance time between the start of the task and the point when they confirmed all necessary changes on all specified objects. At the end of the study, participants were asked to describe their experience of FlashFormat, give suggestions for improvements, and fill out a questionnaire.

We recruited 12 participants (8 male and 4 female, with average age 25; PB1–PB12) from our research institute. All were familiar with PowerPoint and fluent in English, and none of them participated in the first study. The same compensation was offered to all participants in this study.

Results

Figure 4 shows the mean performance time for each number of objects with the two techniques. Times in the manual editing condition varied in the range of 90–100 seconds, whereas FlashFormat times were constant at about 40 seconds. Two-way repeated measure ANOVA revealed a significant main effect for technique ($F_{(1,11)} = 45.5, p < .0001$, $\eta_p^2 = .81$). The interaction between technique and task was also significant ($F_{(2,22)} = 4.40$, $p < .05$, $\eta_p^2 = .29$). The post-hoc analysis confirmed that the FlashFormat technique was significantly faster than the manual editing technique ($p < .0001$). This quantitative result demonstrates substantial improvements over existing editing tools and methods for global restyling, supporting our hypothesis HB-3.

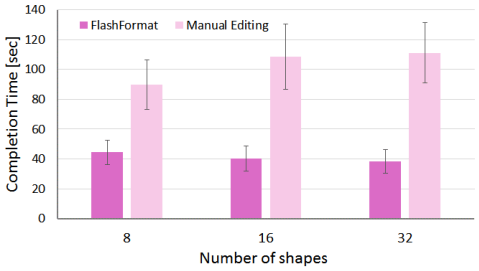


Figure 4. Mean performance time in the second study. The error bars represent 95% Confidence Intervals.

Questions	Mean response (*SD*)
I could always give appropriate examples to reach my desired end state.	5.1 (1.6)
I could anticipate the effects of FlashFormat before actually doing it.	4.6 (1.4)
It is necessary to anticipate the effects of FlashFormat to successfully use the tool.	5.8 (1.2)
It is annoying to repeatedly undo unwanted effects from FlashFormat.	4.3 (2.1)
I would prefer to use FlashFormat rather than make repetitive changes one-by-one.	6.5 (0.8)

Table 5. Responses of the post-experimental questionnaire. (1: strongly disagree – 7: strongly agree)

Table 5 presents the results of the questionnaire. Participants responded positively to their experience with FlashFormat, unanimously agreeing that FlashFormat was preferable to manual editing. As with StyleSnap, many participants commented on how it would support their regular authoring practices, especially with regard to diagram formatting. We further examined the qualitative study data to understand the reasons for these positive results.

Qualitative Analysis

In the pre-task session, participants exhibited a tendency to give one example, repeatedly FlashFormat-all until they reached the point of over-generalization, undo, and repeat. Thus, the participants did not follow the 'golden rule' of giving *more, and more diverse examples*, despite being reminded of it after each undo action. However, as the system repeatedly failed to produce their intended generalization from a single example, participants began giving more examples. At first, these were typically two examples given on the first two slides where they were applicable. However, we observed a gradual shift to a more systematic selection of diverse examples. Not only did participants learn to follow the golden rule through experience, they also learned what it means for examples to be diverse and how to give enough of them:

"Doing things automatically is good but sometimes I need to find out the differences of the shape, for example this is red and this is orange so I know I have to give two examples, one on the red and another on the orange, to let the system know that I want to change all the shapes no matter what the color is." (PB2)

"If you want to change the styles for all slides, you have to review all the slides first. For me it should be: first review all the slides, and then pick up all the different parts… different stuff in the same parts you want to change, and then change the different properties, and go ahead and FlashFormat and it should work." (PB8)

Overall, this suggests that repeated feedback from applying the LGG of self-selected formatting examples is sufficient to promote self-discovery of the optimum example-giving and generalization strategy. Our questionnaire results also support this, showing that participants agreed that they could provide examples and anticipate the effect. Thus, we concluded that HB-1 and HB-2 are also supported.

Nevertheless, the learning process could still benefit from additional guidance. One participant mentioned how before starting new examples, they *"always forget to press the button"* (PB12). Another participant described how, while giving examples, they *"sometimes feel lost about what to do next – should I choose more examples, or should I apply FlashFormat first?"* (PB1). Finally, several participants reported the need for better feedback, before and after applying FlashFormat-all, about which objects would be changed or had been changed already. Suggestions for improvements included highlighting the scope of objects that would be changed next, highlighting the differences of objects that share some attributes with the selected object to guide example selection, learning from multi-object examples where the relative changes are significant, and removing the need to press a start button. The latter could be done either showing a history of recently edited attributes that will be applied or suggesting multiple transformations from a single example.

Even without the suggested changes, the approach was assessed favorably against alternatives. Compared with the PowerPoint Format Painter, it was found to *"work more efficiently"* (PB3), to be *"much faster"* (PB12) and to be *"much more powerful"* (PB4) because it can *"work globally"* (PB4) and *"between slides"* (PB5) without overwriting all attributes (PB4). FlashFormat also *"has some features Slide Master does not provide"* (PB7), such as the ability to work on groups of objects that do not share the same location. In this respect, FlashFormat also surpasses the restyling power of StyleSnap, in which object groups are formed only by shared extents.

OVERALL DISCUSSION

We presented two complementary tools for automating global changes that can improve the visual consistency of slide decks. From the design and evaluation of these two tools we now synthesize limitations, lessons, and future work.

One limitation of the current work is our focus on the design of slide visuals only, and not of the underlying presentation material [23] or narrative structure [31]. While the balance of presentation preparation time should arguably be in favor of content and story, any time saved on styling slide visuals could conceivably be transferred to these other activities.

Another limitation is our sole use of PowerPoint for corpus analysis and prototyping. Since all slides and slideware are structurally similar, we expect our proposed solutions to generalize to other slideware and their associated slides. As for applications to other domains (e.g., vector-based graphical editing), this has not yet been demonstrated.

As we have already noted, StyleSnap does not preserve size and spacing constraints within a slide. Constraint-based reasoning, as used for single layout beautification [41], could provide a solution. Similarly, a limitation of FlashFormat is that imperceptible differences in attribute values are still viewed as differences by the system, leading to potential mismatches between user expectations and action outcomes.

Finally, we have evaluated our two tools independently rather than as a single system, such that we may evaluate their individual value. We hope to combine the functionality of the two tools into a single system in future work, taking into account the lessons discussed next.

Lesson 1: Suggest generalizations from single edits

One of the advantages of StyleSnap over FlashFormat is that users can confidently edit all of the objects in a position group at once rather than having to provide several examples first. One of the advantages of FlashFormat over StyleSnap is that object groups can be formed from any set of shared attributes, not just edge positions. Future work should investigate how to achieve both high predictability and flexibility from single examples. This could be achieved by suggesting multiple possible transformations after each edit, or suggesting snapping results for attribute values beyond edge positions (e.g., to create small, consistent sets of colors and font sizes) after each object selection. Users could thus make progress by incrementally applying such suggestions.

Lesson 2: Support state preservation as well as propagation

In both StyleSnap and FlashFormat, there were times when the user already had an example of their desired end state but were forced to recreate it for the benefit of the tool. In StyleSnap, this was because the snapped object groups represented the modal values of the clustered edges rather than the extent of a specified object. In FlashFormat, this was because only the edited attributes of an object contribute to the inferred example, and none of the other existing attributes. Future work should explore how to give examples of both the "change to" and "keep as" variety on a per-attribute basis, without requiring an enumeration of all object attributes.

Lesson 3: Show the scope of prospective changes

In StyleSnap, the real-time feedback from the combination of object highlights and the Selected slide section gave confidence to the user about the scope of their changes before and as they were making them. However, this feedback also created visual noise for dense slides and caused scrolling of the slide list for large numbers of selected objects. As literature suggests the importance of feedback in this type of systems [29], future work should explore alternative feedback strategies for tools like StyleSnap and FlashFormat.

Lesson 4: Incorporate design patterns and principles

StyleSnap makes the layout of objects consistent across slides, but it does not give any guidance about the desirability of those layouts. Similarly, FlashFormat can make large-scale changes easily, but provides no feedback about the desirability of those changes (e.g., considerations of the contrast between text and its background image following a

global change to caption color). Future work should explore how to resolve aesthetic issues through assisted layout and styling that considers factors such as visual balance [25] and mood [21]. Supporting consistency, not just within sets of user-created visuals but also with external design patterns and principles, remains a significant research challenge.

REFERENCES

1. Abela, A.V. (2008). Advanced presentations by design. Pfeiffer.
2. Ali, K., Hartmann, K., Fuchs, G. & Schumann, H. (2008). Adaptive layout for interactive documents. SmartGraphics'08, 247-254.
3. Amershi, S., Fogarty, J. & Weld., D. (2012). Regroup: interactive machine learning for on-demand group creation in social networks. CHI'12, 21-30.
4. Baudisch, P., Cutrell, E., Hinckley, K. & Eversole, A. (2005). Snap-and-go: helping users align objects without the modality of traditional snapping. CHI'05, 301-310.
5. Bier, E.A. & Stone, M.C. (1986). Snap-dragging. SIGGRAPH'86, 233-240.
6. Blackwell, A.F. (2002). First steps in programming: A rationale for attention investment models. Human Centric Computing Languages and Environments, 2002.
7. Borning, A., Lin, R.K.H & Marriott, K. (2000). Constraint-based document layout for the Web. Multimedia Syst. 8(3), 177-189.
8. Cypher, A. (1991). EAGER: programming repetitive tasks by example. CHI'91, 33-39.
9. Drucker, S.M., Petschnigg, G. & Agrawala, M. (2006). Comparing and managing multiple versions of slide presentations. UIST'06, 47-56.
10. Duarte, N. (2008). Slide:ology: The art and science of creating great presentations. O'Reilly Media.
11. Edge, D. Savage, J. & Yatani, K. (2013). HyperSlides: dynamic presentation prototyping. CHI'13, 671-680.
12. Fails, J.A. & Olsen Jr., D.R. (2003). Interactive machine learning. IUI'03, 39-45.
13. Fernquist, J., Shoemaker, G. & Booth, K. S. (2011). "Oh snap"–helping users align digital objects on touch interfaces. INTERACT'11, 338-355.
14. Frisch, M., Kleinau, S., Langner, R. & Dachselt, R. (2011). Grids and guides: multi-touch layout and alignment tools. CHI'11, 1615-1618.
15. Frisch, M., Langner, R. & Dachselt, R. (2011). Neat: a set of flexible tools and gestures for layout tasks on interactive displays. ITS'11, 1-10.
16. Graf, W.H. (1998). Constraint-based graphical layout of multimodal presentations. Readings in intelligent user interfaces, Morgan Kaufmann, 263-285.
17. Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX: results of empirical and theoretical research. Human mental workload, 1(3), 139-183.
18. Hastie, T., Tibshirani, R. & Friedman, J.J.H. (2001). The elements of statistical learning. Springer.
19. Horvitz, E. (1999). Principles of mixed-initiative user interfaces. CHI'99, 159–166.
20. Jacobs, C., Li, W., Schrier, W., Bargeron, D. & Salesin, D. (2003). Adaptive grid-based document layout. SIGGRAPH'03, 838-847.
21. Jahanian, A., Liu, J., Lin, Q., Tretter, D., O'Brien-Strain, E., Lee, S.C., Lyons, N. & Allebach, J. (2013). Recommendation system for automatic design of magazine covers. IUI'13, 95-106.
22. Kurlander, D. & Bier, E. (1988). Graphical search and replace. SIGGRAPH'88, 113-120.
23. Liu, Y., Edge, D. & Yatani, K. (2013). SidePoint: a peripheral knowledge panel for presentation slide authoring. CHI'13, 681-684.
24. Lok, S. & Feiner, S.K. (2001). A survey of automated layout techniques for information presentations. SmartGraphics'01, 61-68.
25. Lok, S., Feiner, S.K. & Ngai, G. (2004). Evaluation of visual balance for automated layout. IUI'04, 101-108.
26. Masui, T. (2001). HyperSnapping. HCC'01, 188-194.
27. Miller, R.C. & Myers, B.A. (2002). Multiple selections in smart text editing. IUI' 02, 103-110.
28. Miller, R.C. & Marshall, A.M. (2004). Cluster-based find and replace. CHI'04, 57-64.
29. Myers, B. A. (1992). Demonstrational interfaces: a step beyond direct manipulation. Computer 25(8), 61-73.
30. Open XML SDK 2.5. http://msdn.microsoft.com/en-us/library/office/bb448854.aspx
31. Pschetz, L., Yatani, K. & Edge, D. (2014). TurningPoint: narrative-driven presentation planning. CHI'14.
32. Raza, M., Gulwani, S. & Milic-Frayling, N. (2014). Programming by example using least general generalizations. AAAI.
33. Reber, R., Schwarz, N., & Winkielman, P. (2004). Processing fluency and aesthetic pleasure: is beauty in the perceiver's processing experience?. Personality and social psychology review, 8(4), 364-382.
34. Reynolds, G. (2012). Presentation Zen: simple ideas on presentation design and delivery. New Riders.
35. Ritter, A. and Basu, S. (2009). Learning to generalize for complex selection tasks. IUI'09, 167-176.
36. Schrier, E., Dontcheva, M., Jacobs, C., Wade, G. & Salesin, D. (2008). Adaptive layout for dynamically aggregated documents. IUI'08, 99-108.
37. Sibson. R. (1973). SLINK: an optimally efficient algorithm for the single-link cluster method. The Computer Journal 16 (1), 30-34.
38. Watanabe, T. & Hanaue, K. (2013). Composition support of presentation slides based on transformation of semantic relationships into layout structure. Multimedia Services in Intelligent Environments, 25, 155-181.
39. Weitzman, L. & Wittenburg, K. (1996). Grammar-based articulation for multimedia document design. Multimedia Systems, 4(3), 99-111.
40. Wigdor, D., Benko, H., Pella, J., Lombardo, J. & Williams, S. (2011). Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. CHI'11, 1581-1590
41. Xu, P., Fu, H., Igarashi, T. & Tai, C-L. (2014). Global beautification of layouts with interactive ambiguity resolution. UIST'14, 243-252.