PolyZoom: Multiscale and Multifocus Exploration in 2D Visual Spaces

Waqas Javed, Sohaib Ghani, and Niklas Elmqvist
Purdue University
West Lafayette, USA
{wjaved, sghani, elm}@purdue.edu

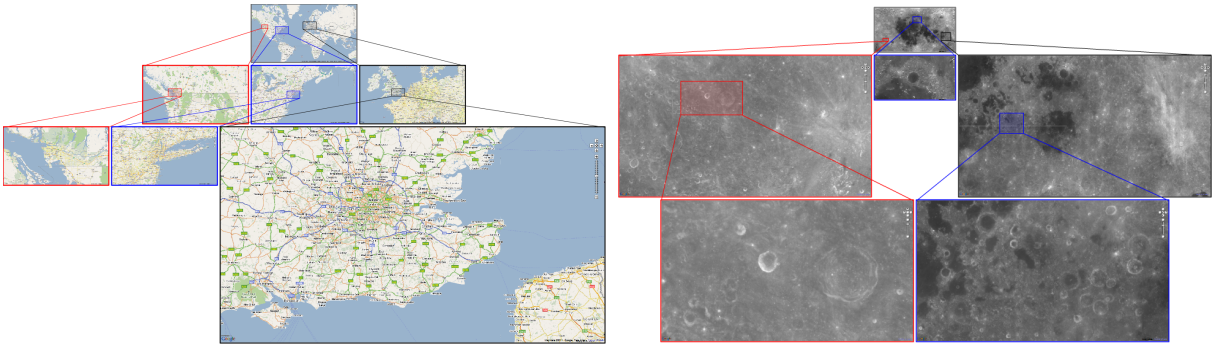


Figure 1. PolyZoom focus hierarchies where the color-coded selections and the connecting lines make the parent-child relationships explicit.

ABSTRACT

The most common techniques for navigating in multiscale visual spaces are pan, zoom, and bird's eye views. However, these techniques are often tedious and cumbersome to use, especially when objects of interest are located far apart. We present the PolyZoom technique where users progressively build hierarchies of focus regions, stacked on each other such that each subsequent level shows a higher magnification. Correlation graphics show the relation between parent and child viewports in the hierarchy. To validate the new technique, we compare it to standard navigation techniques in two user studies, one on multiscale visual search and the other on multifocus interaction. Results show that PolyZoom performs better than current standard techniques.

Author Keywords

Multifocus interaction; comparative visualization; maps; navigation; visual exploration; visual analytics; interaction.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Interaction styles*; I.3.6 Computer Graphics: Methodology and Techniques—*Interaction techniques*

CHI'12, May 5–10, 2012, Austin, Texas, USA.

INTRODUCTION

A *multiscale visual space* has different visual representation depending on the zoom level used to view the space [15]. Navigation in such multiscale visual spaces is becoming increasingly important for many everyday tasks. For example, Google Maps is essentially a large, two-dimensional, and multiscale geospatial dataset. The same could be said about many visualizations, such as zoomable treemaps, large adjacency matrices, or high-resolution scatterplots. However, multiscale navigation has been shown to be a difficult problem because of the phenomenon known as *desert fog* [12], where the immediate environment in the viewport is devoid of navigational cues due to its multiscale nature. For example, in Google Maps, seeing continents on the world map will not help the user in finding a specific park in a city. Zooming into the map to search for the target will automatically cause a loss of overview. However, maintaining an awareness of the full space is necessary for effectively exploring the space, otherwise the navigation process can easily become tedious [5].

We present PolyZoom, a navigation technique for two-dimensional multiscale visual spaces that allows users to iteratively build a hierarchy of focus regions, thereby maintaining their awareness of multiple scales of the visual space at the same time (Figure 1). Using PolyZoom, users can progressively narrow down the search space by creating regions of increasing magnification into areas of interest on the visual space. This allows the user to be aware of the space at different scales, so they can easily backtrack if they make a mistake during the visual exploration. In addition to this, PolyZoom also supports multifocus interaction [2]

where several regions of interest can be compared side by side while preserving their spatial relation and context.

We have implemented PolyZoom using Adobe Flash. The application lets the user to smoothly navigate in a large and multiscale visual space using the PolyZoom technique, as well as using standard pan and zoom. Our implementation currently supports four different visual spaces: the Google Maps world map, the NASA Universe dataset, a Lunar dataset, and a Martian dataset.

The extra spatial awareness provided by PolyZoom comes at the cost of smaller map viewports and suboptimal use of screen space. To study the impact of this on visual exploration, we performed two user studies comparing the new technique to standard pan and zoom. In the first study, participants were asked to perform a multiscale navigation task that required them to search for a particular target using navigational cues at different levels of scale. In the second study, we studied the multifocus capabilities of the technique by asking users to compare potential targets to a specific source object. Despite the extra space required for the PolyZoom technique, results from both studies indicate a performance improvement for the new technique.

BACKGROUND

Here we review general techniques for navigating in 2D spaces, highlighting issues that make navigation difficult.

Common Navigation Techniques

Scrolling is one of the standard techniques for navigating large workspaces, but it can be cumbersome when the space to explore is large. Igarashi and Hinckley [9] proposed a method of speed-dependent automatic zooming (SDAZ) to automatically vary the zoom level depending on the zoom rate. Ishak and Feiner presented content-aware scrolling [10] to vary the direction, speed, and zoom during scrolling based on content properties. Despite these improvements:

Scrolling has little multiscale capability and requires considerable effort when exploring large 2D workspaces.

Panning allows users to directly manipulate the position of the viewport on the visual space, while *zooming* changes the viewport's magnification level (i.e., viewport space size). Since the original infinitely zoomable canvas proposed in Pad [15], there has been a large amount of work in this area. Furnas and Bederson present the space-scale diagram [5] as a method for understanding zoom and pan actions as paths through space-scale. Multiple studies show that combining zooming and panning is more efficient than only panning [5, 21]. However, panning and zooming alone is insufficient:

Pan and zoom for a large visual space means giving up either overview or detail, and can be ineffective [3, 5].

Overview+detail techniques use multiple windows to present both magnified and bird's eye views of the space. The overview window shows the entire workspace while the detail window shows the current focus. Hornbæk and Frøkjær [8] found that overview and detail techniques are easy to use and can outperform pan+zoom and fisheye views for some tasks. Of particular interest to our work is DragMag [22]—discussed in more detail below—which is an overview+detail technique where the user directly drags a focus region on the visual space to control an associated magnification window. In general, overview and detail techniques have been shown to be effective in navigating large spaces [7, 13]. However, overview+detail has problems:

Overview+detail divides user attention between viewports, and scales poorly to high magnification ratios.

Focus+context techniques combine a focus region shown in great detail integrated with the surrounding context, typically using distortion. Magnifying lenses [1] and fisheye views [3] are commonly used focus+context techniques. Rubber sheet stretching [19] is another model of distorting 2D space. Elmqvist et al. [2] propose a space-folding technique called Mélange that combines different parts of the workspace. Despite the benefits achieved through focus+context techniques:

Focus+context is unstable [6], scales poorly to large spaces [8], and the focus transition is nonintuitive [16].

Multiscale Navigation

Multiscale navigation is defined as spatial navigation in a multiscale visual space, i.e. a space that has different visual representation depending on the magnification used to view the space. Multiscale representations are important in many domains such as cartography (where small-scale maps are generated through the abstraction of large-scale datasets), visualization (examples such as zoomable adjacency matrices, treemaps, and data cubes are all multiscale spaces), and even common documents (which exhibit both macro- and micro-level structure). The original Pad [15] provided an infinitely zoomable multiscale space, and introduced the concept of semantic zooming for navigating such structures. However:

Multiscale navigation is difficult due to missing navigational cues at different levels of scale ("desert fog" [12]).

Multifocus Interaction

Multifocus interaction [2] provides a way to simultaneously view and compare multiple regions of interest while exploring large visual spaces such as geographical maps. Below we review prior work in this domain.

Split-screen techniques divide the screen into multiple viewports that each show a portion of the full workspace, and has been adopted for multifocus interaction in the past. Plaisant et al. [18] used a split-screen for exploring large time series data. Shoemaker and Gutwin [20] combine the split-screen approach with dynamic focus creation that automatically define focus regions as the interaction points move further apart. However, these and other techniques have the following characteristic in common:

Split-screen techniques support multifocus awareness, but provide little spatial relations between viewports and generally do not allow for creating cascading viewports.

Many of the focus+context techniques can also be used for multifocus interaction. Techniques like fisheye views [3, 20], rubber sheet [19], and space folding [2] support multiple

focus regions. However, there are issues with such methods:

Multifocus+context techniques do not permit creating focus hierarchies, which is needed for multiscale awareness.

Javed and Elmqvist [11] propose a multifocus overview+detail technique called *stack zooming* that lets the user build a cascaded hierarchy of focus regions. The technique supports most requirements, however, it has a limitation:

Stack zooming is designed for one-dimensional visual spaces (e.g., time series), and its adaptation to two-dimensional visual spaces (e.g., maps) is non-trivial.

POLYZOOM: TECHNIQUE

The PolyZoom technique was designed for exploring multiscale 2D spaces. We summarize our design goals as follows:

- **Multiscale awareness:** Most overview+detail techniques are not effective when navigating large multiscale spaces, since large differences in magnification between overview and detail make navigation impractical. We aim to provide awareness of the visual space at multiple scale levels.
- **Multifocus awareness:** Many tasks require viewing several regions simultaneously, e.g., for side-by-side comparison [2]. We thus want to support multiple foci.
- **No distortion:** Spatial distortion can be confusing because it introduces non-linear elements into the display, elements that are typically not visually stable under translation [6]. We thus want to avoid spatial distortion.
- **No overlap:** We want the technique to avoid overlapping viewports and instead efficiently fill the available space.

The PolyZoom technique lets users build focus hierarchies by selecting regions of interest in a viewport, thereby spawning a child viewport at higher magnification. This gives the user awareness of multiple levels of scale simultaneously. As shown in Figure 2, three focus regions (A, B, and C) have been created on the visual canvas. Each new viewport forms a parent-child relation where the new viewport becomes the child of the parent viewport in which the selection was made. The new viewport can further act as a parent for multiple new viewports by performing the same selection operation. For example, in Figure 2, the viewports A and B, children of the main viewport, are the parents of viewports D and E, respectively. This parent-child viewport relation provides not only overview at multiple scale levels, but also facilitates backtracking while navigating in a multiscale visual space.

Our approach of letting users create focus regions that spawn magnification viewports is reminiscent of the DragMag technique [22], where a magnifying glass metaphor is used to control a zoom window on top of a base window. However, PolyZoom takes the basic DragMag concept further by allowing the user to create hierarchies of zoom windows (i.e., focus regions within zoom windows), thereby producing a multiscale navigation technique. Furthermore, because PolyZoom also supports creating multiple children for each viewport, it inherently supports *multifocus interaction* [2, 20], the capability of comparing multiple focus regions side-by-side. Unlike previous multifocus techniques, such as space folding [2] and multiple fisheye views [20], it does so without using distortion in the display.

The three main design aspects of PolyZoom are (1) the layout algorithm, which dictates the positioning of different viewports on the screen; (2) the viewport size management algorithm, which manages the size of different viewports; and (3) correlation graphics, which provide explicit linking between parent and child viewports. In the following subsections we will discuss each of these in detail.

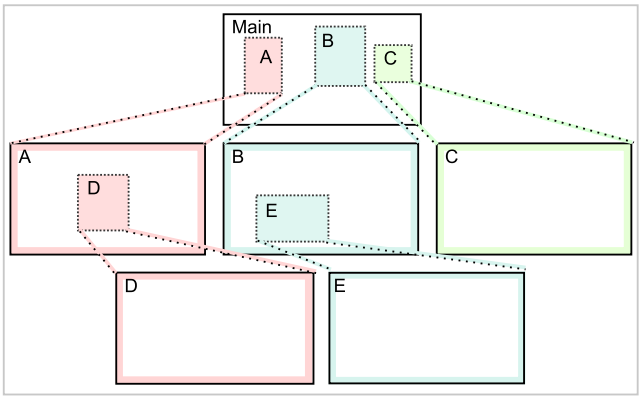


Figure 2. General layout mechanism for PolyZoom. Color-coded zoom areas and corresponding colored frames show parent-child relationships, and visual lines make the relations explicit.

Layout

For the PolyZoom technique, we aimed for a layout that is easy to understand, allows effective side-by-side comparisons at a given level of the focus hierarchy, and is logical and predictable based on the underlying tree structure. Based on these requirements, we use a stack-based algorithm to lay out a focus hierarchy as multiple viewports on the screen. This is done by splitting the vertical space so that each consecutive level in the focus hierarchy is allocated a vertical segment, starting with the focus root at the top of the space. Next the horizontal space in each segment is divided and allocated as viewports to the focus regions on that level in the hierarchy. This layout generates a visual tree structure that makes it easy to relate parent and child viewports, thereby allowing the user to keep track of the context of any focus region all the way to the root of the focus hierarchy. Figure 2 shows an example layout where viewports A, B, and C are children of the main viewport, and viewports D and E are the children of viewports A and B, respectively.

The layout algorithm restricts the vertical positioning of viewports based on the parent-child relation, but makes no assumption on their horizontal positioning. This means that focus regions for a level in the hierarchy can be reordered in a vertical segment, e.g., for side-by-side comparison.

The layout also makes each viewport the same size. For most visual spaces, such as for maps and images, the aspect ratio between width and height is fixed. This means that no dimension in a focus region should be stretched or squashed. For these situations, because PolyZoom allows the user to freely select 2D focus regions, the viewports may need to

show some additional area of the visual space outside the selection in order to preserve the aspect ratio. This is done by fitting a rectangle of the correct aspect ratio to the selection (which most likely does not have the correct aspect ratio). This is only necessary if the underlying visual space is sensitive to varying aspect ratio. Another approach is to restrict the aspect ratio of the focus area during selection.

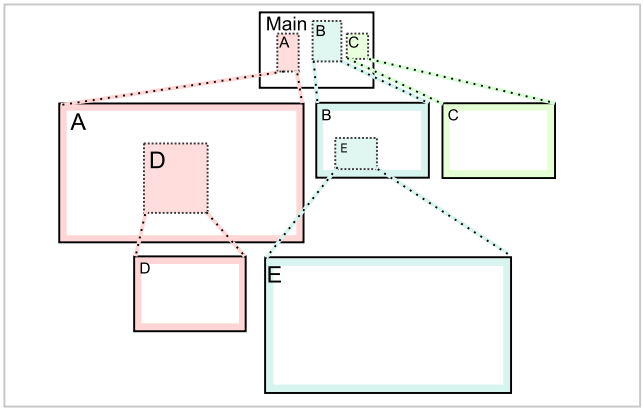


Figure 3. Viewport size management mechanism for PolyZoom. Two windows A and E are selected as active viewports, causing them to be allocated a larger part of the available view space.

Viewport Size Management

So far we have kept viewport size constant when performing the layout of the focus hierarchy in the visual space. However, viewport size plays an important role for the effectiveness of PolyZoom: if viewports become too small as the focus hierarchy grows, it will be difficult to see details in each individual viewport. On the other hand, not all viewports are important all the time, and this will change as the user navigates in the visual space. Hence, dividing the view space equally among all the viewports is not always optimal.

As an alternative, we provide an interest-based viewport size management algorithm that dynamically changes the sizes of individual viewports while keeping their aspect ratio constant. The algorithm allows users to dynamically flag the viewports that are currently of interest as being *active* (or even to let the active viewport be determined by the mouse pointer). Active viewports are assigned a magnification factor M that captures how much larger an active viewport will be compared to a passive one (if $M = 1$, this algorithm reverts to the equal-sized viewport layout algorithm above).

The algorithm proceeds by finding which of the vertical or horizontal dimensions of the focus hierarchy requires the most amount of space. For the vertical axis, this amounts to summing up each level of the hierarchy, where a level has value M if there is at least one active viewport in the level, and 1 if there is none. For the horizontal axis, we select the level with the maximum sum, where the sum of a level is computed by counting each active viewport as M, and each passive one as 1. Finally, we compare the horizontal and vertical sums and select the maximum as the deciding factor (the intuition is that the axis with the maximum sum will require the most display space). This factor is then used to determine both width and height of passive and active viewports based on the magnification factor M.

Figure 3 shows an example of applying this algorithm. Viewports A and E are active. Based on the magnification ratio $M = 2$, the passive viewports are given considerably less space. In this example, the vertical sum is $2M+1$ (dominant) and the horizontal is $M+2$ (for level 2 in the hierarchy).

Correlation Graphics

For maintaining the full context of viewports in the focus hierarchy, we want to make the relationship between parent and child viewports explicit. Prior work [11] in this field uses correlation graphics to achieve this. We adopted this idea for PolyZoom as follows:

- **Color-coded selection areas:** The selection areas in a viewport are rendered as color-coded and semi-transparent rectangles in the viewport the selection was made. This makes it easy to identify the extent of selected space that corresponds to a particular viewport in its parent window. For example, in Figure 2, three regions A, B, and C have been selected in the main viewport. The selection regions are highlighted using three different colors.

- **Color-coded viewport frames:** We use the same color-coding for the frames around child viewports to indicate the relation to the selection region in the parent (Figure 2).

- **Correlation links:** We use explicit correlation links as color-coded lines that connect the selection region in the parent to the child viewport (Figure 2). These are helpful in the presence of a large number of viewports, but may cause visual clutter. For this reason, we only link the base of a focus region to the top of each child viewport.

POLYZOOM: SYSTEM

We built a web-based prototype implementation of PolyZoom for multiscale and multifocus navigation in 2D multiscale visual spaces. Our implementation supports the Google Maps dataset, NASA Universe, a Lunar dataset, and a Martian dataset. Even though our prototype is tailored to these datasets, there is nothing in the technique that prevents it from being implemented for other datasets, such as Bing Maps, DeepZoom[1], or general 2D visual spaces.

Our implementation (Figure 1) starts with a single main viewport that displays the complete visual space (i.e., for Google Maps, the whole world). Users can pan and zoom normally in this viewport just like in a normal Google Maps implementation, i.e., using dragging and the mouse wheel or the Google Maps navigation widget. In addition to this standard functionality, we also allow users to specify focus regions by left-clicking on the map to start a rubber band rectangular selection, and left-clicking again to finish the selection. This creates a new focus region that becomes a child of the main viewport. The selected area in the main viewport is drawn as a color-coded and semi-transparent rectangle. This new child viewport has the exact same functionality as its parent; specifically, it can have its own children if the user

[1] `http://www.microsoft.com/silverlight/deep-zoom/`

performs a focus region selection inside its borders. Furthermore, each viewport can naturally have more than one child. The system also supports all three correlation graphics described above to maximize the user's spatial awareness.

The interaction design of navigating parents and children has different alternatives depending on the application. In our implementation, navigating in a child viewport will update the size and position of its corresponding selection region in the parent; analogously, moving the selection area of a child in a parent viewport is the same as panning the child itself. A child viewport is specified in relative coordinates to its parent, so panning or zooming a parent will automatically pan or zoom the child a corresponding amount. Finally, parents are immutable from the viewpoint of a child, so children viewports are restricted to the bounds of the parent, thereby preventing interaction in one child to affect other siblings.

EVALUATION

PolyZoom provides spatial awareness simultaneously at multiple different levels of scale. However, this comes at the cost of reduced viewport size for each individual focus region. In addition, as can be seen in some of the figures in this paper, laying out a hierarchy of focus regions with a fixed aspect ratio means that the entire screen cannot be fully utilized. In other words, PolyZoom has both potential advantages and disadvantages over the standard pan and zoom operations that are common when navigating in spatial datasets such as Google Maps.

To better understand these tradeoffs, we performed two user studies comparing PolyZoom to standard pan and zoom interaction. Our study designs were influenced by the multiscale search operationalization proposed by Pietriga et al. [17]. Thus, the first study focused on multiscale search, and required participants to navigate in a hierarchy of visual cues on different levels of scale. The second study was aimed at the multifocus aspects, and required participants to compare foci in different locations on the visual space.

Both studies were performed back to back and with the same participants. However, the order of performing a study was balanced between participants. In this section, we discuss common aspects of both studies, whereas in subsequent ones, we describe their unique details.

Participants

We recruited 12 participants (11 male, 1 female) from the student population at our university (average age 22.2, median age 23). Participants were all self-selected volunteers, had normal or corrected-to-normal vision, were not color blind (self-reported), and were paid $10 upon completing a full experimental session. No particular skills were required other than basic knowledge of operating a computer.

Apparatus

The experiment was conducted on a standard 3 GHz dual-core desktop computer with 4 GB RAM and running Microsoft Windows XP. The computer was equipped with a standard two-button mouse (with mouse wheel), a keyboard, and a 19" LCD monitor set to 1280×1024 resolution. Participants only used the mouse to interact during the trials. The experimental application was maximized on the screen.

Navigation Technique (T)

We used Navigation Technique T as a factor for both studies:

- *Standard Pan and Zoom (S):* Navigation in map windows with all of the basic navigation techniques supported by Google Maps. Participants were able to create multiple map windows, and had access to the standard Google Maps overview for each window.

- *PolyZoom (P):* Navigation using the PolyZoom navigation technique; no standard overview viewport.

In the context of Pietriga et al.'s operationalization of multiscale search [17], the *Standard Pan and Zoom* technique (*S*) that serves as our baseline and comparison is equivalent to the "Pan-Zoom + Overview" technique that is most efficient in their experiment, with the difference that we use the standard overview window provided by the Google Maps API. On the other hand, Nekrasovski et al. [14] found that the presence of an overview had no impact on multiscale search, so we think this small discrepancy will have minimal impact.

Software and Dataset

We used our PolyZoom implementation in Adobe Flash/Flex (described above) as the experimental software using the Google Maps standard dataset. This allowed us to use both standard pan and zoom, as provided by the Google Maps API, as well as the PolyZoom technique as discussed above. Depending on the Technique factor, we restricted the use of PolyZoom—participants could still pan and zoom using the normal Google Maps user interface in both conditions.

In Study 2, we asked participants to compare two different regions on the map. In a real setting, a user may solve this task using two or more Google Map windows viewed side-by-side. To support this strategy, we implemented our map windows for the standard pan and zoom option as internal frames in a multiple document interface (MDI). We chose a MDI design (as opposed to creating separate browser windows) so that we were able to instrument the windows for the purpose of control and measurements for the user study.

For the standard pan and zoom condition, tasks started with a single map window, but participants were able to create additional windows simply by clicking a button. Each window supported basic operations such as resizing (by dragging on the lower-right corner of the window), bringing to top (by clicking anywhere on the window), and closing (by clicking on an X in the title bar). Beyond clicking on windows directly, users could switch between windows by clicking on their titles in a taskbar implemented inside our MDI interface. Pressing Shift-Tab also cycled between windows.

For both navigation techniques, viewports could be navigated using all of the basic navigation techniques supported by Google Maps: pan (left-drag), zoom (mouse-wheel), navigation widget, etc. In the PolyZoom condition, participants could also use the PolyZoom technique as described above, but were not allowed to create multiple windows.

Procedure

Upon starting an experimental session, participants were first allocated to a study order (i.e., Study 1 or 2 first). For each study, the experiment administrator described the overall task. The participant was then assigned to an order of techniques for the study, which we blocked on.

At the start of each technique block, participants were asked to complete a set of training trials. During this training session, the administrator demonstrated how to operate the technique and how to use it to solve the task. Participants were then asked to solve training trials on their own. They were allowed to ask questions during the training trials (not during real trials). Training continued until the participants indicated that they were ready to proceed to the timed trials.

A trial started when the participant clicked on a button in a dialog box that interleaved each trial. This dialog box served as an intermission between trials. Clicking on the button started the timer. Participants were instructed to complete the task as quickly as possible. The means of ending a trial (stopping the timer) was different for the two studies:

- For **Study 1**, the trial ended when the user had found and clicked on the red balloon that served as the target.
- For **Study 2**, the trial ended when the user had entered a target name they thought was correct and clicked "OK".

At the end of each study, participants were asked to provide comments and to give 5-point Likert scale ratings on how efficient, how easy, and how rewarding the different techniques were to use.

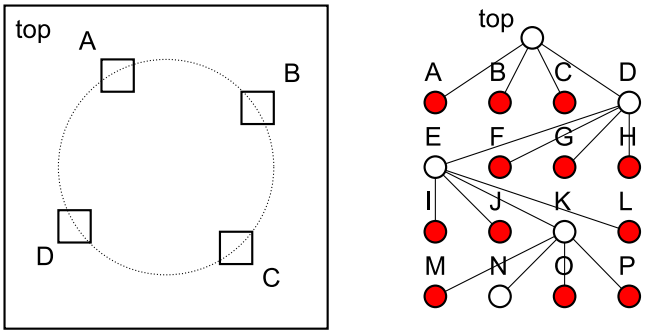


Figure 4. Top level view of the cue hierarchy (left) and complete sample visual cue (right). Red nodes represent dead ends, white nodes are correct choices that give rise to new options. The bottom white node (N) represents the target. The correct path is D, E, K, and N.

STUDY 1: MULTISCALE VISUAL SEARCH

Our first user study was designed to test the basic multiscale navigation functionality of PolyZoom in comparison with the standard pan and zoom provided in Google Maps (the winning technique in the experiment by Pietriga et al. [17]).

Task

The intuition behind our scenario in this study was that multiscale search consists of knowledge at several different levels of scale. To mimic this multiscale knowledge, we asked participants to navigate a hierarchy of visual cues, in the form of identical squares. This would emulate situations where a user knows a little about the target at different zoom levels, e.g., "it is a city near a coast", but several alternatives may have to be pursued to determine which one is correct.

Starting from the top view (shown at the start of a trial), and at each subsequent level in this hierarchy, four visual cues were shown as squares on the visual space. The visual cues form a hierarchy, where three of the four visual cues are dead ends, and only one is the correct and contains four new visual cues in turn. This parent-child relationship between correct visual cues continues for a given number of levels (depending on an experimental factor, see below).

Inside a visual cue, the squares representing the new visual cues were equally spaced, with a random starting offset, on the periphery of an invisible circle centered in the middle of the space. The radius of this circle was equal to one third of the parent cue's width, and the width of a square representing a visual cue is one tenth of the width of the parent. Figure 4 gives an overview, and Figure 5 gives an example.

The cues at different levels of the hierarchy were made visible only when a user is at a zoom level greater than or equal to the zoom level associated with that cue hierarchy level. When zooming into a visual cue representing a dead end, the contents of the square would turn a transparent red (Figure 5(b)) to indicate this fact. When zooming into a visual cue at the bottom of the hierarchy, the final target—a red balloon—would appear (Figure 5(d)). The trial ended (stopping the timer) when the participant clicked the balloon.

To simulate the effect of noticeable change in a multiscale visual space with a change in scale, there was a difference of three zoom levels between two adjacent levels of the cue hierarchy. This difference was also sufficient to ensure that at a time, only one of the visual cue at the previous levels of cue hierarchy is visible (Figure 5(b) and (c)).

Nevertheless, it should be noted that the visual cue hierarchy is *not* a complete tree, where dead ends are only unveiled at the bottom level. We chose to "short-circuit" incorrect search branches immediately by indicating that they are dead ends as soon as the participant zoomed into the visual cue. If we had used a complete tree, PolyZoom would likely benefit because it would offload the participant from having to remember each zoom decision at each node in the tree. The choice not to do this was made to keep the comparison fair. Because this design was in effect (thus minimizing the amount of backtracking arising from errors), we also chose *not* to control for random chance (which we do in Study 2).

Study Design

We used a full-factorial within-participants design:

	12	participants
×	2	**Navigation Technique** T (S, P)
×	3	**Hierarchy Level** L (3, 4, 5)
×	4	repetitions
	288	Total trials (24 per participant)

The L factor modeled the number of levels in the visual cue hierarchy, and we included it to study the effect of naviga-

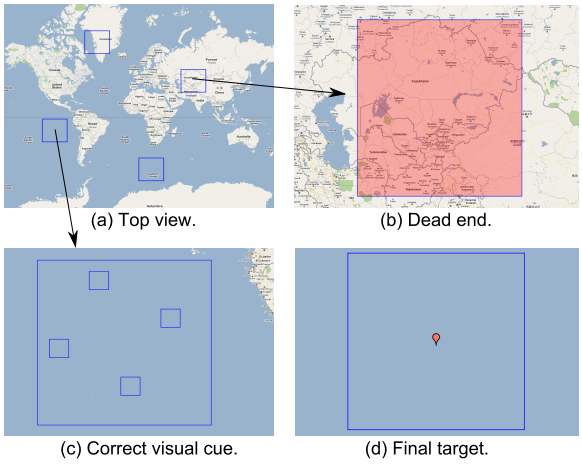


Figure 5. Scenario for Study 1. (a) View at the start of a trial; (b) Visual cues that represent dead ends are colored red when zoomed in; (c) The correct visual cue yields the next level in the hierarchy when zoomed in; (d) Zoom-in on the last level of the correct visual cue, where the target becomes a red balloon that needs to be clicked to finish the experiment.

tional depth on multiscale navigation. The deeper the hierarchy, the more stress will be placed on the navigation techniques: more zoom stacks in PolyZoom, and more backtracking for standard zoom and pan.

We organized the trials in blocks based on technique. Block order and the order of visual cue hierarchy levels within each block were randomized across participants to counteract learning effects. During the experiment we collected the time it took the participants to complete a task. Completion times were averaged for the four repetitions.

Hypotheses

H1 *P will be faster than S.* We believe that PolyZoom's increased multiscale awareness will favor the technique.

Results

We followed common statistical practice to improve the robustness of our completion time data by removing outliers (trials outside two standard deviations of the average) and averaged between the four repetitions per participants. The omitted trials were all above the average, and likely resulted from interface mistakes. We then analyzed the completion time measurements using a repeated-measures analysis of variance (RM-ANOVA, normality and equality of variances assumptions valid). We found a significant main effect of Technique T on completion time ($F(1,11) = 7.33, p < .0071$). Figure 6(a) shows completion times per technique and hierarchy level; the average times were 27.47 (s.d. 5.80) seconds for PolyZoom (P), and 29.39 (s.d. 6.13) seconds for standard pan and zoom (S). This constitutes an approximate 6.5% speedup for PolyZoom over standard pan and zoom, which is a small but non-negligible value.

Not surprisingly, there was also a significant main effect of Hierarchy Level L on completion time ($F(2,22) = 51.76, p < .0001$). Completion time was roughly linear with the number of levels in the visual cue hierarchy. Pairwise comparison of levels using a Tukey HSD showed that levels were significantly different ($p < .05$) with completion times ordered $3 < 4 < 5$. However, we found no significant interaction between T and L. This signifies that there is no difference in how the techniques manage increasing scale.

Participants were able to create multiple windows in the standard pan and zoom condition, but few did so: only in 10 out of the 144 trials for the S condition did participants create additional windows, and even in those situations, the maximum number of windows ever created was 2. Creating a second window is not an advantageous strategy for this task, but the low number of instances this happened means there should be only a very small impact on our results.

Subjective Feedback

Subjective ratings are shown in Figure 7 (left); differences were significant (Friedman test, $p < .05$) except for reward. In their comments also, participants found PolyZoom useful for the given task. One of them stated "[PolyZoom] is cool," and another wrote "PolyZoom was very useful."

Discussion

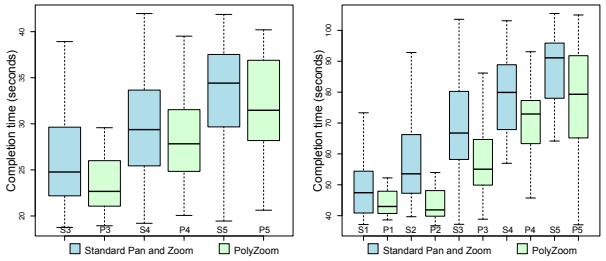
Our results confirmed hypothesis H1: participants performed the multiscale search task 6.5% faster using PolyZoom than using standard pan and zoom. While the speedup is fairly small, this is nevertheless a significant result given the optimality of pan and zoom as indicated by related work—for example, Pietriga et al. [17] found that pan and zoom performed better than two advanced lens-based techniques, and Hornbæk and Frøkjær [8] showed that overview+detail was superior to panning and fisheye views.

The experimental application was maximized on the screen at all times, which meant that both techniques had access to the same, constant amount of display space. The explanation for these results is likely that PolyZoom provides more awareness of the visual space than pan and zoom with a single overview. Our findings indicate that the impact of viewports becoming successively smaller, as well as suboptimal use of display space, did not decrease performance for PolyZoom, at least not in situations covered in our experiment.

The solution strategy that users adopted with PolyZoom was interesting and worth describing: for each level in the visual cue hierarchy, the user would create *one* focus region centered on one of the four visual cues. This new viewport would immediately allow the user to determine whether this was a dead end or not, and if it was, the user would simply move the selection area in the parent to the next visual cue. In other words, participants only ever had a single viewport open at each zoom level. For pan and zoom, on the other hand, the participants were forced to continually pan and zoom the single viewport to focus on each visual cue.

The question is whether and how these results generalize, and we think there are both qualitative and quantitative answers to this question. On a qualitative level, we believe that it is easy to see the usefulness of PolyZoom for general multiscale navigation, and we can foresee the technique being used for even larger and more complex visual spaces than those tested in our experiment. On a quantitative level, we

found no significant interaction between technique and hierarchy level, which seems to indicate that PolyZoom's performance did not degrade as the number of levels in the visual cue hierarchy increased. On the other hand, since PolyZoom keeps every step of the navigation process visible, it is clear that there *does* exist a limit to how far it will scale (although it is always possible to restart PolyZoom on a new position in the space, foregoing the existing focus hierarchy). However, it should be noted that in our evaluation, the condition $L = 5$ used the full 23 zoom levels of the Google Maps dataset, which clearly is a large and realistic multiscale visual space.



(a) Completion times (S1). (b) Completion times (S2).

Figure 6. Performance metrics for both Study 1 and Study 2 as a function of the level L and discovery order D, respectively.

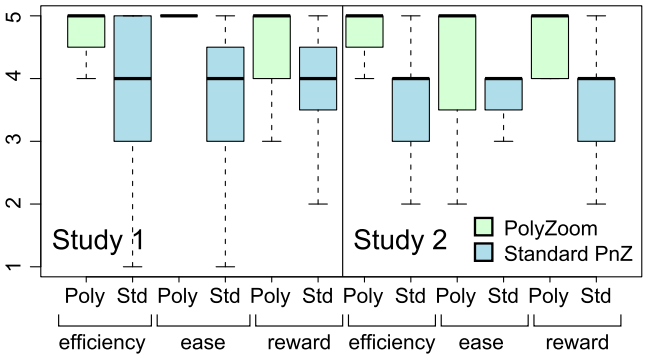
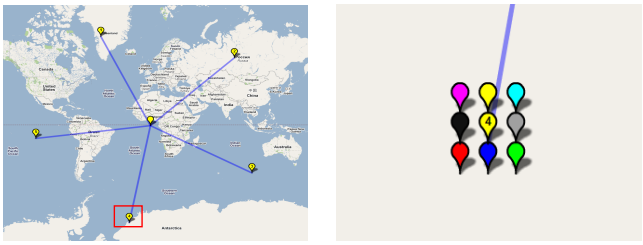


Figure 7. Subjective ratings for both studies (Likert 1-5 scale).



(a) Overall view. (b) Detail view.

Figure 8. Scenario for Study 2. (a) Overall trial view. The target in the center is the source, and the targets at the spokes of the circle are destinations. (b) Detail view of target 4 (indicated in the overview with a red rectangle) showing the 3 × 3 grid of colored balloons (only visible at maximum zoom) used to represent a target.

STUDY 2: MULTIFOCUS COMPARISON

Our second study was intended to explore the multifocus capabilities of PolyZoom. The novel technique provides a way to manage a hierarchy of viewports, whereas the de facto standard for looking at multiple regions in the same visual space is to open two or more windows and compare them side by side. We wanted to see whether the ever-shrinking viewports of PolyZoom would actually result in better performance than the multiple window approach.

Task

The motivation for the task in our second study was to capture situations when a user is trying to compare a particular region in a visual space with a set of candidate regions to find the one that matches. For this reason, we designed this task as a multifocus comparison task.

A trial setup consisted of a *source target* placed at the center of the visual space, and five potential *destination targets* arranged with equal spacing on the periphery of a virtual circle centered at the source. The source was connected to all destinations by a straight line. The task consisted of determining which of the five destination target exactly matches the appearance of the source. Figure 8(a) gives an example.

Actually determining the appearance of a target—source or destination alike—required navigating to the maximum zoom level in the visual space for the target to be unveiled. A target is a 3 × 3 grid of colored balloons where the center balloon is always yellow (Figure 8(b)). The colors used were red, green, blue, black, yellow, purple, cyan, and grey. The source target has a unique configuration of colors, and one of the destination targets—the *actual target*—has the exact same configuration. All other destination targets have the same color configuration but with two randomly swapped colors. This was done to ensure that all destination targets would be similar to the source, precluding matching on just a part of the source's color configuration.

Completing the trial consisted of finding the actual target and inputing its number (shown in both overview and detail views, see Figure 8) into a text field and clicking a button.

Study Design

We used a full-factorial within-participants design:

	12	participants
×	2	**Navigation Technique** T (S, P)
×	5	**Discovery Order** D (1, 2, 3, 4, 5)
×	2	repetitions
	240	Total trials (20 per participant)

We wanted to counterbalance the impact of random chance in the navigation, so we introduced a factor D for which order the participant would discover the actual target. In other words, while we told the participants that the program randomly selected one destination target to be the actual target prior to the participant starting the trial, this was not actually the case. Instead, the application counted the number of destination targets as they were unveiled (i.e., viewed at the maximum zoom level), and made the n:th destination target the actual target. With D taking on all values $\{1,2,3,4,5\}$, and with two repetitions, it is easy to see that all participants twice encountered all orderings of when they found the actual target. This design mimics the counterbalancing done

by Pietriga et al. [17] and controls the impact of participants being "lucky" and choosing the right target every time.

We again organized the trials in blocks by technique. Technique order was balanced and discovery orders were randomized to counteract learning effects. During the experiment we measured time and correctness. To detect guessing, if a participant answered without ever having unveiled the actual target, we counted that as a wrong answer. This was also true for the last target, so participants were not able use elimination (we informed participants of this fact).

Hypotheses

H2 *P will be faster than S.* We think that PolyZoom's explicit multifocus support will outperform pan and zoom with access to multiple windows; no predictions on correctness.

Results

As in Study 1, we removed outliers and calculated the average across all repetitions of a trial. We then analyzed the completion times using a repeated-measures analysis of variance (as before, all assumptions were valid). We found a significant main effect of technique T on completion time ($F(1,11) = 28.10, p = .0001$). Figure 6(b) shows the completion times per technique and order; the average times were 60.7 (s.d. 17.6) seconds for PolyZoom, and 68.31 (s.d. 19.1) seconds for standard pan and zoom, an 11% speedup.

Not surprisingly, the discovery order D had a significant main effect on completion time ($F(4,44) = 46.08, p < .0001$). The effect was roughly linear, as expected, and the pairwise differences in completion time between orders were significant (Tukey HSD, $p < .05$). We found no interaction effects between the two factors T and D, however.

We analyzed the correctness using logistic regression, but found no significant main or interaction effects of T or D on this metric. In general, correctness for Study 2 was high, with more than 90% of all trials being answered correctly.

For the number of windows, all participants created at least two windows when solving the task (there was no single trial with less than two windows). This tells us that participants were using an appropriate strategy, i.e., putting one window on the source target and then creating at least one window for comparing destination targets. Two was also the most common number, used in 32% of trials. These findings are also consistent with our informal observations from the study.

Subjective Feedback

Differences in subjective ratings (see Figure 7 (right)) were significant (Friedman test, $p < .05$) except for ease of use. Like for Study 1, comments given by the participants were in favor of PolyZoom; for example, "PolyZoom [was] better," and "[the standard pan and zoom] technique did not help."

Discussion

Our results again confirm our hypothesis H2: participants were 11% faster when using PolyZoom than when they were using standard panning and zooming with two or more map windows. This is significant given that spawning multiple windows, each supporting pan and zoom, is the current baseline for existing tools such as Google Maps. In fact, our experimental platform made creating new map windows trivial by providing an explicit button for this, whereas in a real window environment you would need to duplicate your browser window to achieve this. In other words, we suspect a lightweight technique such as PolyZoom that incorporates multiple viewports may compare even better.

The reason for PolyZoom performing better than standard pan and zoom is fairly straightforward: PolyZoom not only makes creating multiple viewports easy, but it also provides an explicit correlation between viewports. We did not evaluate this latter characteristic in Study 2, but we think it is easy to see that giving the user an awareness of the spatial relation of viewports may be highly useful in many situations.

It is worth noting that we did *not* compare PolyZoom against existing multifocus interaction techniques such as space-folding [2], multiple fisheyes [20], or rubber sheet interaction [19]. However, none of these techniques support multi-level awareness, and they all rely on visual distortion which is visually unstable and can cause hunting effects [6]. While it would be interesting to compare PolyZoom directly to these techniques, we leave this for future work.

Finally, in terms of generalizing these results, we again give both qualitative and quantitative arguments. For a qualitative argument, we believe that PolyZoom will also generalize to other, and potentially more complex, multifocus tasks than the ones studied here. For example, our task only involved comparing two regions, but it is easy to see that PolyZoom could be used to compare three, four, or more regions. As for the quantitative argument, we again saw no significant interaction between technique and discovery order. In other words, participants were not degrading in performance as a function of how late the actual target came in the order of visitation. While this will again clearly not scale indefinitely, we still think that it indicates that our results will generalize outside the context of our experimental setup.

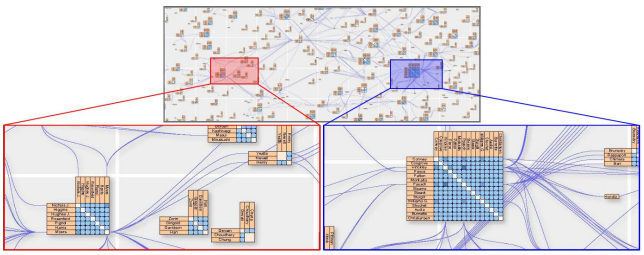


Figure 9. PolyZooming in the 20 years of UIST diagram.

POLYZOOM: BEYOND MAPS

While both of our studies used the multiscale geographical world maps through Google Maps, none of the actual tasks actually made use of the geographical data itself. We used the Google Maps example because it is familiar to our participants, and it also represents the actual state of the art of what people use today when interacting with maps.

However, as we have already established, PolyZoom can be used for any 2D visual space, not just geographical maps.

Figure 1 shows a visual exploration session in a Lunar dataset where the user is comparing the different craters on the lunar surface. Similarly, we envision applying PolyZoom to a variety of multiscale spaces defined by visualization techniques, including large adjacency matrices, multiscale hierarchies, and multiresolution high-dimensional data. Figure 9 shows an example of zooming in the large UIST co-authorship graph created by Henry, Dragicevic, and Fekete (`http://www.aviz.fr/gallery/uist20years_v2.jpg`). Our findings in this paper easily confirm the general utility of PolyZoom for multiscale spaces and we foresee much advances for other datasets.

CONCLUSIONS AND FUTURE WORK

We have presented PolyZoom, a multiscale and multifocus navigation technique in which users progressively build hierarchies of focus regions that are rendered on the screen simultaneously. We have implemented PolyZoom for Google Maps, NASA Universe, and other multiscale datasets, and we use this implementation to compare the technique to pan and zoom in two controlled experiments involving human subjects. Our findings show that PolyZoom is superior for both multiscale search as well as multifocus comparison. In the future we plan to continue to improve these tasks through interest-based, topology-aware, and semantic methods.

REFERENCES

1. C. Appert, O. Chapuis, and E. Pietriga. High-precision magnification lenses. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 273–282, 2010.
2. N. Elmqvist, N. Henry, Y. Riche, and J.-D. Fekete. Mélange: Space folding for multi-focus interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1333–1342, 2008.
3. G. W. Furnas. Generalized fisheye views. In *Proceedings of the ACM Conference on Human Factors in Computer Systems*, 16–23, 1986.
4. G. W. Furnas and B. B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 234–241, 1995.
5. C. Gutwin. Improving focus targeting in interactive fisheye views. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 267–274, 2002.
6. K. Hornbæk, B. B. Bederson, and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Transactions on Computer-Human Interaction*, 9(4):362–389, 2002.
7. K. Hornbæk and E. Frøkjær. Reading of electronic documents: The usability of linear, fisheye, and overview+detail interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 293–300, 2001.
8. T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 139–148, 2000.
9. E. W. Ishak and S. Feiner. Content-aware scrolling. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 155–158, 2006.
10. W. Javed and N. Elmqvist. Stack zooming for multi-focus interaction in time-series data visualization. In *Proceedings of the IEEE Pacific Visualization Symposium*, 33–40, 2010.
11. S. Jul and G. W. Furnas. Critical zones in desert fog: Aids to multiscale navigation. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 97–106, 1998.
12. V. Kaptelinin. A comparison of four navigation techniques in a 2D browsing task. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 282–283, 1995.
13. D. Nekrasovski, A. Bodnar, J. McGrenere, F. Guimbretière, and T. Munzner. An evaluation of pan & zoom and rubber sheet navigation with and without an overview. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 11–20, 2006.
14. K. Perlin and D. Fox. Pad: An alternative approach to the computer interface. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, 57–64, 1993.
15. E. Pietriga and C. Appert. Sigma Lenses: Focus-context transitions combining space, time and translucence. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1343–1352, 2008.
16. E. Pietriga, C. Appert, and M. Beaudouin-Lafon. Pointing and beyond: an operationalization and preliminary evaluation of multi-scale searching. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 1215–1224, 2007.
17. C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. LifeLines: Visualizing personal histories. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 221–227, 1996.
18. M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss. Stretching the rubber sheet: A metaphor for visualizing large layouts on small screens. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, 81–91, 1993.
19. G. Shoemaker and C. Gutwin. Supporting multi-point interaction in visual workspaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 999–1008, 2007.
20. J. J. van Wijk and W. A. A. Nuij. Smooth and efficient zooming and panning. In *Proceedings of the IEEE Symposium on Information Visualization*, 15–22, 2003.
21. C. Ware and M. Lewis. The DragMag image magnifier. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, 407–408, 1995.