Hierarchically Animated Transitions in Visualizations of Tree Structures

David Guilmaine
École de technologie supérieure
Montreal, Canada
david.guilmaine@gmail.com

Christophe Viau
École de technologie supérieure
Montreal, Canada
christopheviau@gmail.com

Michael J. McGuffin
École de technologie supérieure
Montreal, Canada
michael.mcguffin@etsmtl.ca

ABSTRACT

We present an experimental comparison of 4 techniques for smoothly animating changes in a radial tree visualization. Two traditional techniques, linear and staged animation, are compared with two novel techniques: a hierarchical animation that proceeds level-by-level through the tree, and a hybrid animation technique that mixes the staged and hierarchical approaches. Users were asked to track changes in nodes of the tree during animated transitions. Results show a significant advantage for the hierarchical and hybrid animation techniques for tracking certain kinds of changes. We then propose guidelines for designing animated transitions. Finally, we present a new popup widget for interactively controlling the progression of an animation that combines advantages of previous widgets.

Categories and Subject Descriptors

I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Viewing algorithms*

General Terms

Human Factors, Experimentation

Keywords

tree visualization, animation, smoothly animated transitions

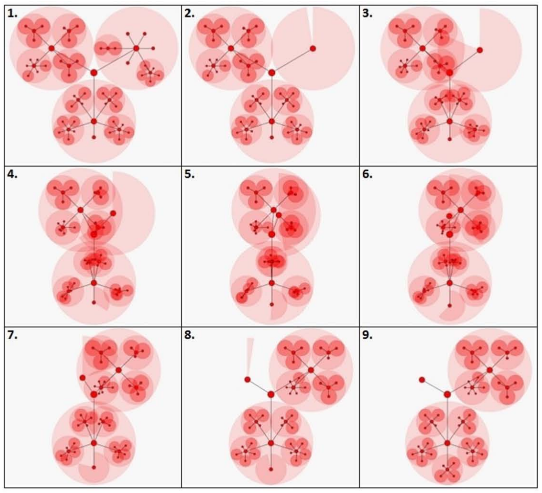
1. INTRODUCTION

Smoothly animated transitions have become increasingly common in interactive visualizations [16, 10, 8]. Recent experiments [10, 19] have shown that when a user changes the view or representation of their data, rather than show a sudden and discontinuous change in visual feedback, it is better to show an animation of the data elements continuously transitioning from their initial state to their new state, as this helps the user better understand the change in the view. An alternative to showing any transition is to show the old and new views (or a sequence of views) side-by-side, as in small multiples [17, 1, 22]. However, small multiples reduce the screen space allotted to each view, and cannot be used if the user wants to see only one view at a time, to maximize the space available for that view.

Although animated transitions are better than no animation, one remaining research question is how to best design these transitions. The simplest design has all changes in the visualization animated simultaneously; we call this kind of transition *linear*. SpaceTrees [16] proposed breaking the transition into 3 stages: a 1st stage where elements are collapsed, a 2nd stage where remaining elements are moved, and a 3rd where elements are expanded. Heer and Robertson [10] found that such *staged* transitions were subjectively preferred by users, but found only modest performance gains with staged transitions compared to linear transitions, and even found that staged transitions were sometimes outperformed by linear transitions.

Staged transitions break the transition into steps based on the kinds of changes to show. An alternative would be to modify the transition based on the structure of the data, or the connections between data elements. For example, Heer and Robertson [10] proposed *staggered* transitions, where the animation of each element is delayed incrementally, according to the ordering of bars in a barchart for example, to reduce inter-element occlusion. We explore a different data-dependent strategy in the case of tree visualizations, and propose breaking the transition into non-overlapping steps based on the hierarchical levels of the tree. Specifically, we investigate transitions that animate changes on each level of a tree separately, which we call *hierarchical* transitions. We report an experiment comparing linear, staged, and hierarchical transitions, as well as a *hybrid* transition technique that mixes the staged and hierarchical approaches. The hierarchical and hybrid transitions yielded a significant advantage over the other techniques in certain cases.

Our hierarchical and hybrid transitions are a potential way to improve the displayed *output* during a transition. As a complementary topic, we also investigated ways of using and improving *input* during a transition. For example, mouse control could be used by the user to slow down or replay a transition, helping the user to better understand the changes occurring. Toward the end of this paper, we discuss the desirable properties of interactive techniques for controlling



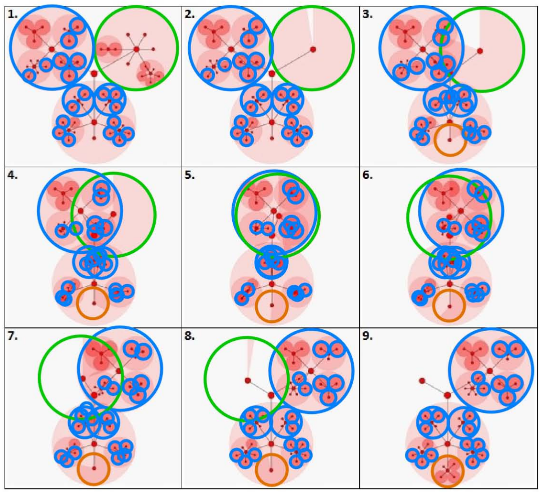


Figure 1: A *linear* transition, where collapsing, permuting, and expanding occur simultaneously. In this example, the north-east subtree in frame 1 (we will call this NE for short) is collapsed; the north-east and north-west subtrees (NE and NW) are swapped (and other, deeper nodes are also permuted); and the south child of the south subtree (which we will call S-S) is expanded. The top version of the figure shows what the user sees. As an aid for the reader, the bottom version highlights collapsing nodes in green, expanding nodes in orange, and permuting nodes in blue.

transitions, and present a taxonomy of widget designs exhibiting these properties.

Our contributions are two new transition techniques (hierarchical and hybrid), an experimental evaluation of these techniques, two new guidelines for designing transitions, and novel widget designs for controlling transitions that are organized in a taxonomy.

2. BACKGROUND

Animation has many applications and benefits in user interfaces [2, 3]. In visualization, animation could be used, for

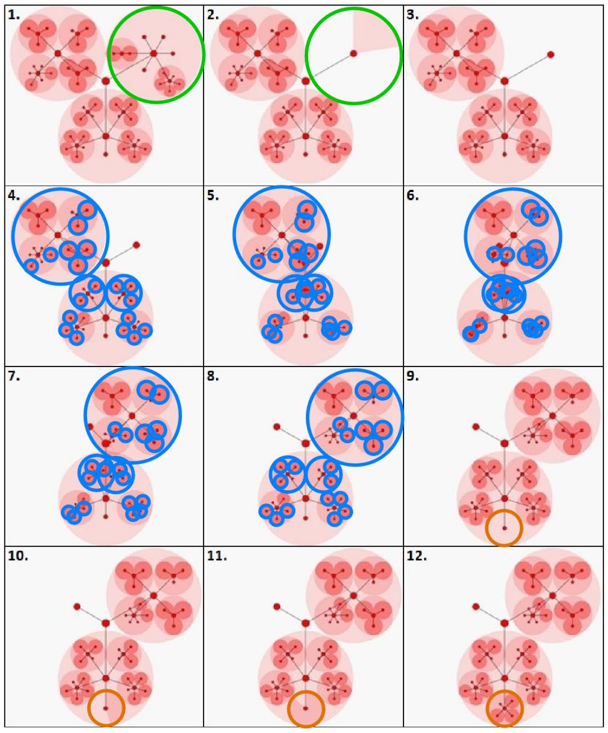


Figure 2: A *staged* transition, which performs collapsing, then permuting, then expanding. Frames 1-3: the NE subtree is collapsed. Frames 4-8: nodes on all 3 levels of the tree are permuted (for example, NE and NW are swapped, and S-NE and S-NW are also swapped). Frames 9-12: S-S is expanded. Highlighting in this figure (and Figures 4, 6) is for the reader's benefit, and is not shown to the user.

example, to shown an additional dimension of a dataset, beyond the dimensions shown in the spatial substrate or in the marks of a visualization. Tversky et al. [21] present evidence that animation may often not be well suited for portraying changes over time when compared to static representations that convey the same information (a recent example agreeing with this is Robertson et al. [17]). However, Tversky et al. do not rule out the appropriateness of animation for "real time reorientations". Indeed, our work focuses on the use of animation for depicting changes in a view of a dataset, in which case a smoothly animated transition is preferable to a sudden, discrete change, as shown experimentally [19, 10].

Breaking an animation into stages [16] has been proposed as a way of better displaying them. Heer and Robertson [10] found that these were preferred by users, but found only modest performance gains with staging, and in some cases staging performed worse than linear transitions. We propose alternative ways of breaking up a transition into steps.

Differentiated transitions [18] are another way of improving transitions, by animating each element in a way that is a function of the type of element, conveying additional information about the element. However, differentiated transitions must be designed according to the kind of data being shown using data-dependent metaphors, and hence may be more difficult to generalize.

Dragicevic et al. [7] experimentally compared constant-speed animations with slow-in/slow-out and other variable speed animations, and found advantages with slow-in/slow-out. Our work instead focuses on the breaking of transitions into non-overlapping steps, with each step played out with constant speed, however our proposed animation techniques could be used in conjunction with slow-in/slow-out.

3. TREE VISUALIZATION

Dozens of visualizations of trees have been catalogued [11]. We decided to focus on radial, nested circle visualizations of trees [5, 20]. For any given static view of such a tree visualization, the radial, nested circle layout is usually more space-efficient than many other tree visualizations, allocating more area to leaf nodes than classical layouts (as shown mathematically in [14]). It is also an easy-to-understand layout. These two qualities make it desirable to users. At the same time, animated transitions that change the positions of nodes in such layouts often create much inter-node occlusion, making it challenging to design transitions that are easy to understand.

We also decided to study transitions that are more complex than those used in SpaceTree [16]. With SpaceTree, a transition occurs whenever the user clicks on a new focal node n, causing certain nodes outside n's subtree to collapse, and then causing nodes to be repositioned, and finally causing nodes under n to be expanded. This kind of transition seems to work very well in SpaceTree because the user only expresses interest in one node at a time, and collapsing and expanding always occur in predictable regions of the tree, outside and within, respectively, n's subtree. In contrast, we were interested in studying more complex transitions where the user cannot predict where collapsing or expanding may occur.

For example, if the user is viewing a tree and, over time, interactively collapses or expands several nodes, and then performs a multi-level undo to return to a previous state, there may be many unrelated nodes that need to be un-collapsed or un-expanded. Complex transitions may also occur if the user wishes to return to a previously bookmarked view, or selects a preset view (for example, sorting all nodes by decreasing size and collapsing all nodes below a size threshold, or sorting all nodes in decreasing recency and collapsing all nodes older than a time threshold). Complex transitions could also occur if a DOI function [9] is used to determine which nodes to expand or collapse, and the user is interactively moving the focal node for the DOI function.

Another scenario where complex transitions may occur is if the tree is a model of a factory, industrial plant, or computer network, and the visualization is used to monitor the system over time, with nodes automatically expanding or collapsing based on activity, events, or problems within them. The children of each node may also be sorted or permuted according to a dynamically computed metric of importance. In such a case, transitions from one state to another may collapse, expand, and/or permute nodes at any level of the tree, in unpredictable ways.

Our software prototype displays a visualization of a tree and can display transitions involving 3 kinds of changes: collapsing a subtree, permuting the children of a node (causing them to be repositioned), and expanding a subtree (revealing its children). The two first transition techniques we implemented were linear (Figure 1) and staged (Figure 2).

Depth

	1	2	…	N
Collapsing	1	1	…	1
Permuting	1	1	…	1
Expanding	1	1	…	1

Linear (1 step)

Depth

	1	2	…	N
Collapsing	1	1	…	1
Permuting	2	2	…	2
Expanding	3	3	…	3

Staged (3 steps)

Collapsing	1	4	…	3N-2
Permuting	2	5	…	3N-1
Expanding	3	6	…	3N

Level-by-stage (3N steps)

Collapsing	1	2	…	N
Permuting	N+1	N+2	…	2N
Expanding	2N+1	2N+2	…	3N

Stage-by-level (3N steps)

Collapsing	1	2	…	N
Permuting	1	2	…	N
Expanding	1	2	…	N

Hierarchical (N steps)

Collapsing	1	1	…	1
Permuting	2	3	…	N+1
Expanding	N+2	N+2	…	N+2

Hybrid (N+2 steps)

Figure 3: Matrices describing different transitions. Columns correspond to depth within the tree, and rows correspond to types of changes. Each transition is broken into steps, and numbers in the cells specify the step at which each cell's change occurs.

3.1 Hierarchical and Hybrid Transitions

There are multiple ways to break up a transition into steps based on the levels in a tree. We developed a matrix notation to unambiguously specify the ordering of changes in different transitions, shown in Figure 3. For example, in a linear transition, all changes occur simultaneously, so there is a single step, and all cells in the matrix contain the number 1. However, in a staged animation, there are 3 steps, and all expanding changes occur at the same time during step 3, hence the cells corresponding to that change in the matrix contain the number 3.

The two middle matrices in Figure 3 show ways of performing staged transition in each level of the tree separately: either performing all 3 stages in the 1st level, then all 3 in the 2nd level, etc. ("Level-by-stage") or performing collapsing in the 1st level, then the 2nd, etc., then permuting in the 1st level, 2nd level, etc., then expanding in each level ("Stage-by-level"). Both of these approaches require a total of $3N$ steps, where N is the depth of the tree. We decided that these approaches required too many steps to be useful: a tree with a mere 3 levels could require as many as 9 steps in a transition.

If we allow collapsing, permuting, and expanding to occur simultaneously, but on each level separately, we obtain a hierarchical transition (bottom left matrix in Figure 3, and Figure 4). To understand the rationale for this technique, consider a parent node p and its children a, b, c, d under-

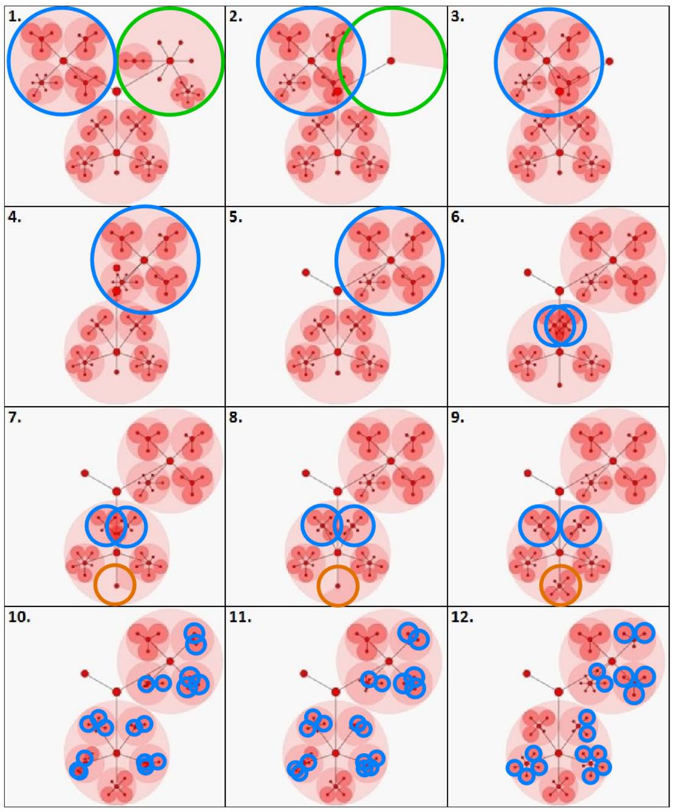


Figure 4: A *hierarchical* transition, where changes occur at level 1, then level 2, etc. Frames 1-5: NE collapses, and NE and NW are simultaneously swapped. Frames 5-9: S-S expands, and S-NE and S-NW are simultaneously swapped. Frames 9-12: nodes on the 3rd level of the node are permuted.

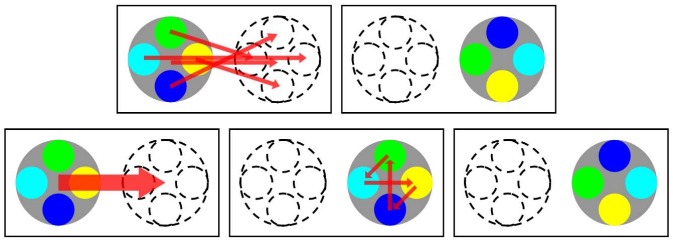


Figure 5: Top row: the displacements of a parent node and its children in a linear transition, in a single step. Bottom row: the displacements of the same nodes in two steps of a hierarchical transition. Dashed circles show the target positions and previous positions of nodes.

going a change that moves p to a new location and also permutes the relative positions of a, b, c, d within p. With a linear transition, each of the five nodes move toward their new final position, probably along non-parallel lines, thus appearing as separate objects to an observer (Figure 5, top row). As surveyed in [6], observers have the ability to track approximately 4 targets simultaneously, however not much more. With a hierarchical transition, one step i would reposition p (with its children), and the next step $i+1$ would reposition a, b, c, d. During step i, the five nodes move together as a group toward p's new location, and during this movement they would be perceived as a single object, possibly creating less load on the observer's perceptual system. During step $i+1$, the children would then move, and have less distance to travel compared to the linear transition because they would already be close to their final positions (Figure 5, bottom row). In a hierarchical transition, this would continue recursively.

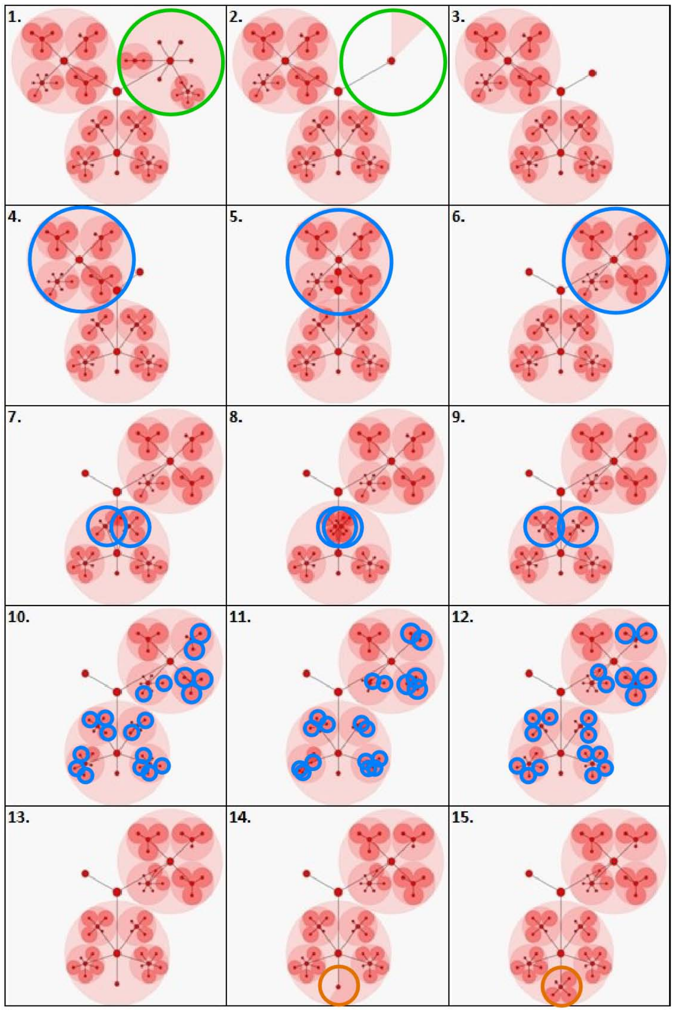


Figure 6: A *hybrid* transition, which collapses nodes on all levels, then permutes on each level, then expands on all levels. Frames 1-3: NE collapses. Frames 4-6: NE and NW are swapped. Frames 7-9: S-NE and S-NW are swapped. Frames 10-12: nodes on the 3rd level of the node are permuted. Frames 13-15: S-S is expanded.

Another way to think about the hierarchical transition is that the ratio of the distance travelled by a node divided by the radius of the node is roughly constant, whereas with a linear or staged transition this ratio increases with the depth of the node. Thus, with a hierarchical transition, as movements become more *detailed* (acting on deeper, more numerous nodes), they act on smaller spatial scales. We hypothesize that this property of hierarchical transitions may help to guide the user's attention from one level to the next.

We also implemented a variant of the hierarchical transition that leverages an idea that motivated staged transitions: it may be advantageous to perform all collapsing up-front, to minimize the number of visible elements during displacements or permutations. Thus, our hybrid technique (bottom right matrix in Figure 3, and Figure 6) collapses nodes on all levels as a first step, then permutes on each level, then finally expands on all levels.

4. EXPERIMENTAL COMPARISON

We compared the four transition techniques using two tasks. The primary task was to track the motion of a single target node that was highlighted at the start of the trial (Figure 7), since users are often particularly interested in changes to a single node. To also evaluate the user's ability to remain aware of other changes in the tree, a secondary task that had to be performed simultaneously was to notice other changes elsewhere in the tree (nodes undergoing collapsing, expanding, or permuting).

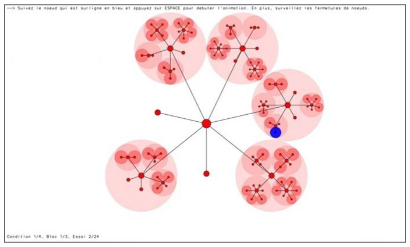

--> Suivez le noeud qui est surligne en bleu et appuyez sur ESPACE pour debuter l'animation. En plus, surveillez les fermetures de noeuds.
Condition 1/4, Bloc 1/3, Essai 2/24

Figure 7: Experimental stimuli.

At the start of each trial, the target node (always a leaf node, on the 3rd level) for the primary task was highlighted. Also, a string at the top of the screen provided instructions for the secondary task, informing the user whether they would need to monitor for collapsing nodes, or expanding nodes, or permuting nodes. (Between 1 and 3 nodes on the 1st level of the tree would undergo the indicated change and need to be noticed by the user for the secondary task.) The user would then hit the spacebar to indicate they are ready, the highlighting would disappear, and the transition would play out, which could permute nodes on any level as well as expand or collapse nodes on the 1st level. At the end of the transition, the user would click on the leaf node they believed to be the target node, completing the primary task. The user would then click on between 1 and 3 nodes on the 1st level that they believed underwent the change for the secondary task. The user would then hit the spacebar to complete the task. Feedback was then displayed to the user to inform them how many correct nodes they clicked on for the primary and secondary tasks, allowing them to adjust their strategy for the subsequent trial.

The experiment was performed on a recently purchased laptop with a 1680×1050 pixel, 15.4 inch screen, and hardware-accelerated graphics. 12 users participated (including 2 women), all university engineering students who use computers 5-15 hours/day, aged 21-45 years of age (average 29). In total, there were 4 conditions (linear, staged, hierarchical, hybrid) × 3 kinds of changes to monitor for the secondary task (expanding, collapsing, permuting) × 3 (for 1, 2, or 3 nodes that underwent a change for the secondary task) × 8 repetitions × 12 participants = 3456 trials.

Warmup trials were performed for each transition technique, to familiarize users with the tasks. However, the design of each transition technique was not explained to participants. The order of presentation of the four transition techniques was counterbalanced using a 4×4 Latin square, and the ordering of trials within each condition was determined pseudorandomly using the same four seeds for each participant. A post-questionnaire solicited subjective preferences.

The tree involved always had 3 levels. The root node could have between 3 and 8 children, and nodes on the 1st and 2nd levels could each have between 2 and 6 children. These numbers were chosen to avoid excessively small nodes. The total time for each transition was always 5 seconds, regardless of the number of steps involved. For the primary (tracking) task, the node to track always underwent a translation of at least 100 pixels. Many of these parameters were chosen following a pilot experiment used for fine tuning.

4.1 Results

There was a total of 864 trials for each transition technique. In the primary task, the number of successful trials for each technique was 619 (linear), 608 (staged), 699 (hierarchical), 653 (hybrid); see Figure 8. ANOVA found that the hierarchical technique significantly outperformed the other techniques ($F_{3,33} = 10.3$, $p < 0.05$). No other significant differences were found. We also measured the time required for users to click on the target node at the end of a trial, and found that with the linear technique, users were significantly slower (1.45 seconds on average for linear, compared with average times of 1.28 seconds or less for the other techniques; $F_{3,33} = 7.1$, $p < 0.01$). We did not ask users to click as quickly as possible on the target node at the end of the trial, nor did we control for the initial cursor position, however the slower times suggests that users hesitated more about where to click following a linear transition, possibly due to the high degree of occlusion that occurs during a linear transition.

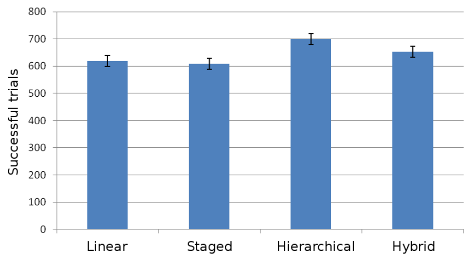


Figure 8: Successful trials for the primary task (tracking a single node).

In the secondary task, the number of successful trials for each technique was 588 (linear), 467 (staged), 580 (hierarchical), 615 (hybrid); see Figure 9. ANOVA found that the staged technique was significantly worse than the other techniques ($F_{3,33} = 22.2$, $p < 0.001$).

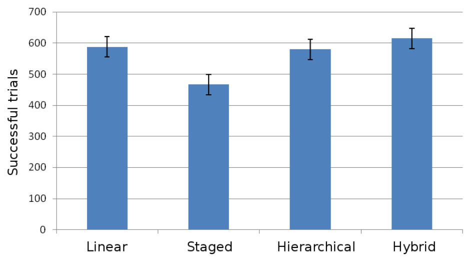


Figure 9: Successful trials for the secondary task (noticing a change in the tree).

Results for the secondary task are broken down according to the type of change to monitor in Figure 10. For secondary tasks requiring that the use identify expanding nodes, the hierarchical technique was significantly worse than the other techniques. When identifying collapsing nodes, the linear technique was significantly better than the other techniques. Finally, when identifying permuting nodes, the hierarchical and hybrid techniques were significantly better than the linear and staged techniques.

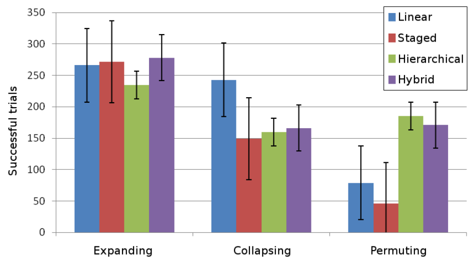


Figure 10: Successful trials for the secondary task, broken down by type of change to notice. Each bar is out of a maximum possible of 288 trials.

The results for expanding and collapsing nodes in the secondary task can be explained by considering which changes are visible toward the end of a transition. We suspect that users spent most of the time during a transition monitoring the primary task's target node, and only toward the end of the transition (when the target node has gotten very close to its final position) does it become easier for the user to allocate attentional resources to check for the location of nodes undergoing changes as part of the secondary task. We noticed that, because of the way we animated collapsing and expanding nodes as pie slices undergoing an angular shrinking or growing (see Figure 1, NE node, frames 2 and 3), in linear transitions these collapsing or expanding nodes are visibly changing right up to the very end of the transition, either as narrow pie slices or as discs missing a narrow slice, respectively. So, such changes are easy to notice toward the end of a linear transition, explaining the high success rates in Figure 10 for linear in expanding and collapsing. However, in hierarchical transitions, the expanding of nodes on the 1st level of the tree occurs at the beginning of the transition, and the collapsing of 1st-level nodes in the three non-linear techniques also occurs at the beginning of the transition, meaning these changes are no longer visible toward the end of the transition, explaining the lower success rates in those cases.

Users ranked the four techniques by preference for the primary and secondary tasks. In the case of the primary task, the results are roughly evenly divided and show no particular tendencies. For the secondary task, however, there seemed to be clearly preferred techniques (Figure 11).

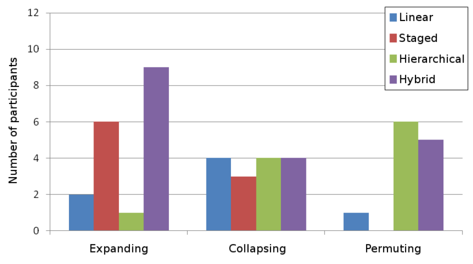


Figure 11: Subjective preferences for the secondary task. Each of the 12 users were allowed to choose more than one "favorite" technique in each case, hence the bar heights can sum to more than 12.

4.2 Discussion

For the primary (tracking) task, hierarchical transitions were significantly better than other techniques, with hybrid transitions a close second. The reason hybrid transitions were slightly worse in this task may be because hybrid transitions involve 2 extra steps, leaving less time for permutations, requiring them to be accelerated.

In the secondary task, when users needed to notice permutations, hierarchical and hybrid transitions were significantly better and also subjectively preferred. However, in the secondary task, noticing expanding or collapsing nodes was best supported by techniques that made these changes visible just before the end of the transition. We therefore propose a change to the hybrid technique that may make it a good candidate for all 3 kinds of secondary tasks: simply stretch out the collapsing of nodes to last all $N + 2$ steps instead of finishing during the 1st step (Figure 12). This way, nodes will still be visibly collapsing just before the end of the transition, possibly improving the performance of this modified hybrid transition in the secondary task where users must notice collapsing nodes.

Depth

	1	2	…	N
Collapsing	1..N+2	1..N+2	…	1..N+2
Permuting	2	3	…	N+1
Expanding	N+2	N+2	…	N+2

Modified Hybrid (N+2 steps)

Figure 12: A proposed modification to the hybrid transition technique. Here, collapsing takes place gradually over all N+2 steps (indicating by the $1..N+2$ in the cells), meaning that the nodes undergoing collapse are still visible (in the form of narrow pie slices) just before the end of the transition.

As discussed earlier, the results in Figure 10 can be explained by considering which changes are still visible at the end of a transition. When designing new transitions, we may assume that the user will often focus on one element in particular, but will also want to remain aware of other kinds of changes during the transition. To help the user, we suggest the following guidelines for designing transitions.

• Changes that are more likely to be important to the user should remaining visible toward the end of the transition. This could be done by performing such changes during the last step of a multi-step transition, or stretching out the change over many steps to include the last step, and/or by highlighting the changes with a color that persists until the end of the transition. (Such color highlighting might even persist with an "afterglow", somewhat like Phosphor [4].)

• Changes that are less likely to be of interest to the user could be performed in earlier steps of the transition.

5. WIDGETS FOR INTERACTIVE CONTROL

Improving the visual display of transitions is one way to make them easier to understand. Another, complementary way, is to give the user interactive control over the progress of animated transitions, allowing the user to traverse multiple transitions when desired, and also to slow down, reverse, or replay a transition to understand all the changes.

We observed that a transition might be invoked with a discrete input action, such as hitting a physical or virtual button (e.g., a "Back" button), or selecting an object (e.g., an element to zoom into). Such discrete actions can activate a transition that plays out at a constant speed. Another example of such discrete input is in [15], where left and right flicks within a radial menu cause layers of a volumetric model to be peeled or unpeeled. Discrete actions have the advantage that they are simple and can be repeated in quick succession, possibly invoking an entire sequence of transitions.

On the other hand, if the user wants to be able to slow down or stop a transition to better understand all the changes taking place, continuous control is appropriate, such as that afforded by a slider widget, or a dragging gesture. For example, dragging has been used for expanding and collapsing multiple levels of a graph (see Figure 14 in [13]).

ScatterDice [8] allows for both kinds of control over transitions. The scatterplot matrix in ScatterDice gives an overview of all views of the data; clicking on a cell in the matrix invokes a constant-speed transition in the main view, while "scratching" (i.e., dragging) on cells allows the user to continuously control the progress of a transition.

We propose using instead a popup widget to control transitions, to benefit from the advantages of popup widgets, namely that they only occupy screen space when in use, and they don't require the user to move toward a menubar or toolbar to invoke them. For maximal flexibility, such a widget should support both discrete input to invoke transitions at constant-speed, and continuous control (dragging) for control over the progress of a transition. For discrete control, we feel that left- and right-flicks are a good choice for many applications, since they can be performed ballistically as soon as the widget is popped up (similar to marking menu gestures [12]) and since left and right naturally map to backwards and forwards in a history of views. Next, there needs to be some way to indicate when the user wants to use continuous control. Figure 13 shows two possibilities: *perpendicular motion* (e.g., upward dragging) could be used, causing a popup slider to appear; or the user could drag and stop over an *activation zone*, causing a slider to expand open. In Figure 13, these two possibilities are crossed with linear and circular sliders. Linear sliders have the advantage that users may be able to drag more quickly to the very end of a sequence of views, whereas circular sliders allow the user to continuously vary the angular gain by moving their cursor closer or further away from the center of the slider.

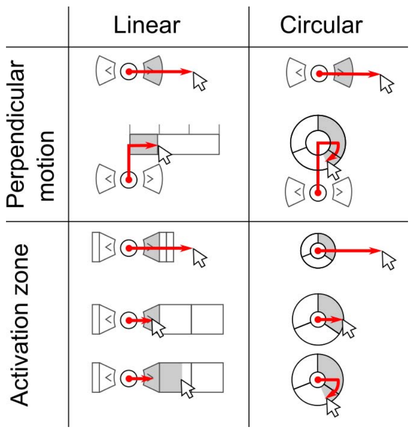


Figure 13: A taxonomy of popup widgets for controlling transitions. Right (or left) flicks invoke transitions that play out at a constant rate, advancing (or backing up) in a sequence of transition states. Each widget also enables the widget to invoke a slider for continuous control over a transition.

A mock-up sketch of the lower-left widget in Figure 13 is shown in Figure 14. We have also implemented a very similar widget for navigating a video (Figure 15).

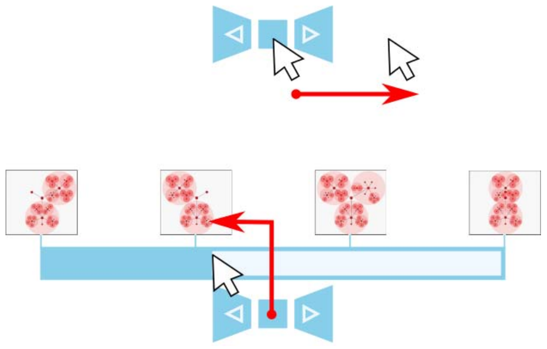


Figure 14: A mock-up sketch of a popup widget for controlling transitions. Top: the user flicks right or left to advance or backup to the next or previous view in a history or sequence of visualization states. Bottom: dragging up causes a slider to appear, afterwhich the user may drag sideways to continuously move forward or backward through transitions to other states. Thumbnails show each state.

6. CONCLUSIONS

Our novel hierarchical transition technique significantly outperformed other techniques in a target-tracking task, with our novel hybrid technique coming in as a close second. Both

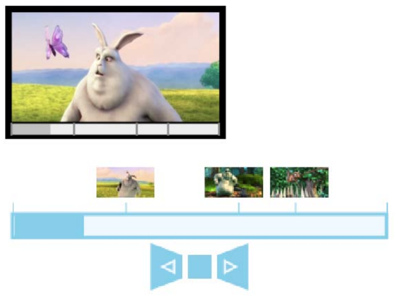


Figure 15: An implemented popup widget for navigating a video timeline. Sideways flicks jump to different chapters in the video. Dragging up and sideways enables continuous navigation within a timeline, with thumbnails corresponding to chapter boundaries. Video frames from `www.bigbuckbunny.org`, © Blender Foundation.

of the new techniques also significantly outperformed existing techniques in a task where users needed to notice additional nodes undergoing permutations, and were also subjectively preferred.

To improve interactive control over transitions, we have also presented designs for widgets that enable both discrete and continuous control over transitions, combining the advantages of previous approaches.

Future work could study if the hierarchical approach taken in this work could be scaled up to larger numbers of elements, and/or applied to non-tree data, such as graph structures (for example, animated nodes that are progressively further and further away from a given focal node).

7. ACKNOWLEDGMENTS

Thanks to Pierre Dragicevic for insightful feedback, and to the participants in our study for their time. This work was funded by NSERC and FQRNT.

8. REFERENCES

[1] D. Archambault, H. C. Purchase, and B. Pinaud. Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE TVCG*, 17(4):539–552, 2011.

[2] R. M. Baecker and I. Small. Animation at the interface, 1990. A chapter (pp. 251–267) in Brenda Laurel, editor, The Art of Human-Computer Interface Design, Addison-Wesley.

[3] L. Bartram. Can motion increase user interface bandwidth? In *Proc. IEEE Conference on Systems, Man, and Cybernetics*, pages 1686–1692, 1997.

[4] P. Baudisch, D. Tan, M. Collomb, D. Robbins, K. Hinckley, M. Agrawala, S. Zhao, and G. Ramos. Phosphor: explaining transitions in the user interface using afterglow effects. In *Proc. ACM UIST*, 2006.

[5] R. Boardman. Bubble trees: The visualization of hierarchical information structures. In *Extended Abstracts of ACM CHI 2000*, pages 315–316, 2000.

[6] P. Cavanagh and G. A. Alvarez. Tracking multiple targets with multifocal attention. *TRENDS in Cognitive Sciences*, 9(7):349–354, July 2005.

[7] P. Dragicevic, A. Bezerianos, W. Javed, N. Elmqvist, and J.-D. Fekete. Temporal distortion for animated transitions. In *Proc. ACM CHI*, 2011.

[8] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE TVCG*, 14(6):1141–1148, 2008.

[9] G. W. Furnas. Generalized fisheye views. In *Proc. ACM CHI*, pages 16–23, 1986.

[10] J. Heer and G. G. Robertson. Animated transitions in statistical data graphics. *IEEE TVCG*, 13(6):1240–1247, 2007.

[11] S. Jürgensmann and H.-J. Schulz. A visual survey of tree visualization. In *Proc. IEEE InfoVis Poster Compendium*, 2010. http://treevis.net/.

[12] G. Kurtenbach and W. Buxton. The limits of expert performance using hierarchic marking menus. In *Proc. ACM CHI*, pages 482–487, 1993.

[13] M. J. McGuffin and R. Balakrishnan. Interactive visualization of genealogical graphs. In *Proc. IEEE InfoVis*, pages 17–24, 2005.

[14] M. J. McGuffin and J.-M. Robert. Quantifying the space-efficiency of 2D graphical representations of trees. *Information Visualization*, 9(2):115–140, 2010.

[15] M. J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proc. IEEE Visualization (VIS)*, pages 401–408, 2003.

[16] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proc. IEEE InfoVis*, pages 57–64, 2002.

[17] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of animation in trend visualization. *IEEE TVCG*, 14(6):1325–1332, 2008.

[18] C. Schlienger, P. Dragicevic, C. Ollagnon, and S. Chatty. Les transitions visuelles différenciées : principes et applications. In *Proc. AFIHM Conférence Francophone sur l'Interaction Homme-Machine (IHM)*, pages 59–66, 2006.

[19] M. Shanmugasundaram, P. Irani, and C. Gutwin. Can smooth view transitions facilitate perceptual constancy in node-link diagrams? In *Proc. Graphics Interface (GI)*, pages 71–78, 2007.

[20] S. T. Teoh and K.-L. Ma. RINGS: A technique for visualizing large hierarchies. In *Proc. Graph Drawing (GD)*, pages 268–275, 2002.

[21] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57:247–262, 2002.

[22] L. Zaman, A. Kalra, and W. Stuerzlinger. The effect of animation, dual view, difference layers, and relative re-layout in hierarchical diagram differencing. In *Proc. Graphics Interface (GI)*, pages 183–190, 2011.