Visualization of Data Changes in 2.5D Treemaps using Procedural Textures and Animated Transitions

Daniel Limberger
Hasso Plattner Institute, Digital Engineering Faculty,
University of Potsdam, Germany

Willy Scheibel
Hasso Plattner Institute, Digital Engineering Faculty,
University of Potsdam, Germany

Jan van Dieken
Hasso Plattner Institute, Digital Engineering Faculty,
University of Potsdam, Germany

Jürgen Döllner
Hasso Plattner Institute, Digital Engineering Faculty,
University of Potsdam, Germany

ABSTRACT

This work investigates the extent to which animated procedural texture patterns can be used to support the representation of changes in 2.5D treemaps. Changes in height, color, and area of individual nodes can easily be visualized using animated transitions. Especially for changes in the color attribute, plain animated transitions are not able to directly communicate the direction of change itself. We show how procedural texture patterns can be superimposed to the color mapping and support transitions. To this end, we discuss qualitative properties of each pattern, demonstrate their ability to communicate change direction both with and without animation, and conclude which of the patterns are more likely to increase effectiveness and correctness of the change mapping in 2.5D treemaps.

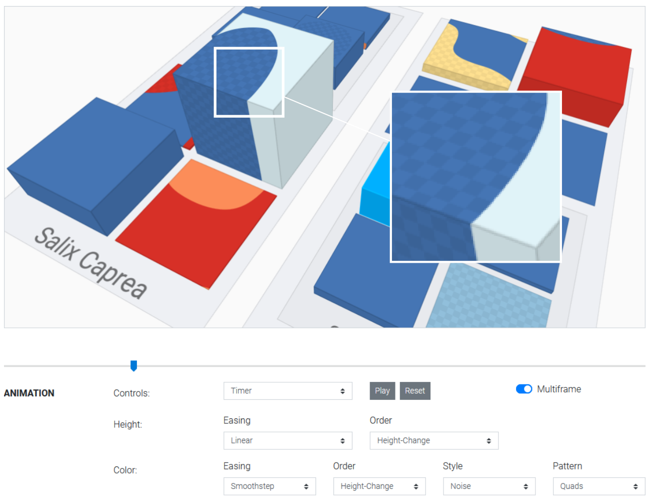
CCS CONCEPTS

• **Human-centered computing** → **Treemaps**; *Information visualization.*

KEYWORDS

visualization, time-variant data, change visualization, 2.5d treemaps, animated transitions, procedural textures, read direction

ACM Reference Format:
Daniel Limberger, Willy Scheibel, Jan van Dieken, and Jürgen Döllner. 2021. Visualization of Data Changes in 2.5D Treemaps using Procedural Textures and Animated Transitions. In *The 14th International Symposium on Visual Information Communication and Interaction (VINCI '21), September 6–8, 2021, Potsdam, Germany.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3481549.3481570



ANIMATION
Controls: Timer · Play · Reset · Multiframe
Height: Easing: Linear · Order: Height-Change
Color: Easing: Smoothstep · Order: Height-Change · Style: Noise · Pattern: Quads

Figure 1: Tiny hierarchical test data set depicted using a 2.5D treemap that allows to transition between subsequent values in time-series data mapped to area, height, and color. A combination of two of our procedural patterns is used to animate changes in subsequent values mapped to color.

1 INTRODUCTION

Treemaps have become a highly expressive tool for visualizing hierarchical, multi-variate data in a variety of domains, including software data [7]. Typically, the data represented has a timestamp and changes over time. Software maps, for example, can represent aspects of a software system for a specific revision or time. We want to enable users to interactively explore correlations in time-varying software system engineering data, i.e., metrics per revision, development budgets, status and progress of defects and issues, team composition, development activity, etc.. An explicit mapping of the difference between two subsequent states to color or height would be the first choice. For 2D treemaps, color patterns such as *two corners*—resulting in a diagonal color gradient—or contrast modifications like *ratio shading* can be used [12]. The time dimension can further be incorporated by distributing multiple maps spatially using small multiples [2, 9]. Provided the evolution (over more that just two states) is already analyzed and available by means of trend data, natural metaphors such as physical-based material degradation, shininess, or glyphs can be an effective tool for communication [15].

We want to assist height and color mappings already in use (*map themes*) and intrinsically emphasize temporal changes, but rely on the user's existing understanding of treemaps. Thereby, the representation of change should visually correspond to the change in data (*visual-data-correspondence*) [4], i.e., the actual difference between two values and the direction of change (or sign of the difference). For this, basic variants of of *data vases* [10], glyphs [3],

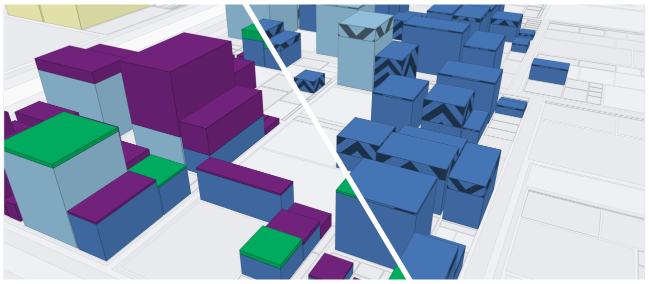


Figure 2: Basic approach for communicating the delta between subsequent values mapped to height: colors (left) and arrows (right) mapped to the delta in height.

or *in-situ templates* [6] can be used. The latter allow for change encoding of up to three visual variables simultaneously, though, ample training is required and issues with reading direction must be considered. A simple but effective use of in-situ templates for display of changes in height is given in Figure 2, one using explicit color mapping (thereby partially occluding the actual color mapping) the other a procedural patterns for direction encoding (using an arrow pattern). Alternatively, animation by means of an interpolation of height and color attributes [1, 5] could be used.

In this work, we combine these two approaches: we utilize procedural texture patterns for more effective animated transitions (Figure 1). Our main contributions are

- the introduction of procedural texture patterns that are suitable for mapping change in color in 2.5D treemaps,
- the idea of primary and secondary patterns for accurate display of change direction when the animation is paused,
- and the evaluation of said patterns w.r.t. their applicability under various constraints and circumstances.

2 PATTERNS FOR ANIMATED TRANSITIONS

We propose seven patterns that can encode the transition between two colors. As these patterns share approaches but also differ in their visual display and characteristics to encode additional state or ensure certain invariants regarding the encoding, we introduce the alternatives and compare them (section 3).

2.1 Procedural Texture Patterns

A pattern describes a mapping of a transition progress for a fragment on a cuboid's surface resulting in a binary per-fragment choice of color. This is done using *signed-distance-functions* [14] and fragment shaders. The patterns differ mainly in their appearance, granularity, and the surfaces they are intended for.

Pillar (vertical). The pillar pattern dissect the surface vertically and assigns the former color and next color to a top share and a bottom share of the pillar, respectively (Figure 3). A transition using this pattern starts with a full pillar using the former color. Then, the next color *grows* from the top and overlays the former color step by step. Thus, after a brief training period, a user should be able to determine the former color and the next color, even for a static display. However, this pattern does not show partial progress on the

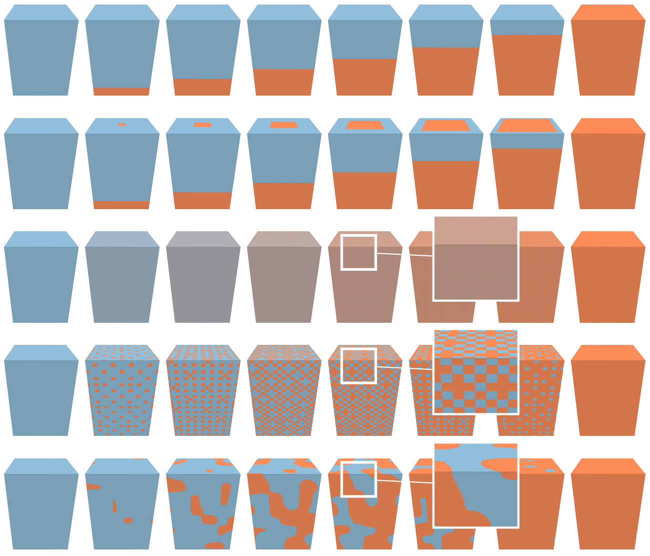


Figure 3: Procedural patterns for changes in color attribute, from top to bottom: pillar, pyramid, dithering, squares, and noise. All of the above are not capable to encode the direction of change ($\Delta_{\text{DirChange}}$) when animation is paused.

top faces. Instead, the color of the top face changes instantaneously with the completion of the transition.

Pyramid (vertical). The pyramid pattern is an extension to the pillar pattern w.r.t. explicit handling of the top faces. The lateral faces use the same encoding. However, the pattern name *pyramid* describes the metaphorical process that is used to derive the pattern: a virtual pyramid is embedded in the cuboid and slowly pushed towards and through the top of it (Figure 3). All cuboid fragments that intersect with the pyramid are assigned the upcoming color. This intersection surface corresponds to the progress of the pillar template for lateral faces and results in a growing rectangle for top faces. As a result, this pattern encodes the current progress of transition on both the lateral faces and the top face.

Noise. The noise pattern uses a 3D noise function [8] and a threshold to discriminate the surface for the two colors. Using such a pattern results in organic-looking surfaces and transition behavior (Figure 3). However, a sensible parameterization of the noise w.r.t. the scaling is challenging. As one approach we propose to approximate the scale factor by the number of nodes in the tree.

Dithering. The dithering pattern encodes the transition using a per-pixel dithering pattern (Figure 3). The dithering is applied in screen space and from a perceptional point of view, this results in a visual blending of the two colors [13]. However, using dithering we assert that only actual colors from a color scheme will be used.

Squares. The squares pattern extend the dithering by using areas larger that one pixel for the dithering raster (Figure 3). As another difference, the pattern is applied in the world space of the treemap and not the raster on the screen. This reduces the visual blurring of colors while providing a transition nonetheless.

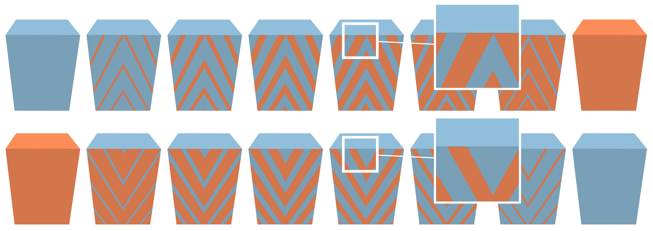


Figure 4: Arrow pattern filling up the color display for increasing (top) or decreasing (bottom) values. The arrows encode value increase or decrease resp., even when paused.

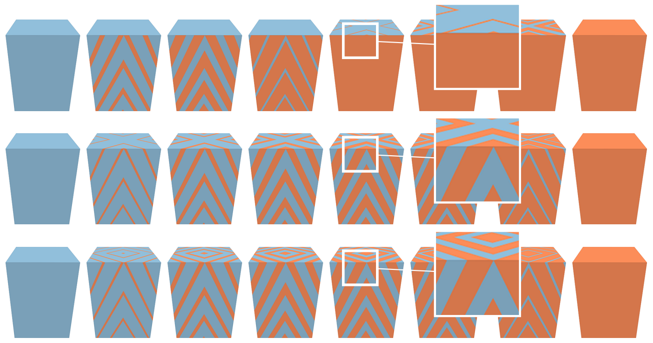


Figure 5: Temporal variants of the arrow-pattern transition for increasing values: lateral face first, then top face (top), or lateral and top faces simultaneously (center and bottom).

Arrows. This pattern is a modulated, two-dimensional *abs* function applied to every lateral face individually. The top faces behaves similar to the pillar pattern. In contrast to the pillar and pyramid patterns, the arrow direction can encode the direction of value change, as the arrows can point upwards or downwards (Figure 4). This, however, prevents effective encoding of direction of animation.

Arrows (full). This pattern extends the arrow pattern by additional arrows on the top. Their direction allows for multiple variants (Figure 5): For one, they can be appended to the transition of the lateral faces. This allows an encoding of direction on top only for a short duration of the transition. Alternatively, the arrows can be animated on the top faces for the whole duration of the transition, whereby two variants with opposite directions are available.

2.2 Pattern Composition and Contours

Invariant to all patterns is that not both the direction of animation and the direction of value change can be encoded at the same time. As further indicator for the direction of animation, we propose a combination of patterns. Thereby, one of the two distinctly colored areas is superimpose by an additional pattern (Figure 6). This allows for direct unambiguous display of change direction, both when paused and animated, as only one state has a second pattern superimposed. This shape of the pattern remains unchanged ($t = 0.5$) during the transition. However, as the overall transition changes the surfaces encoding former and next color, the application of the secondary pattern will vary during the transition.

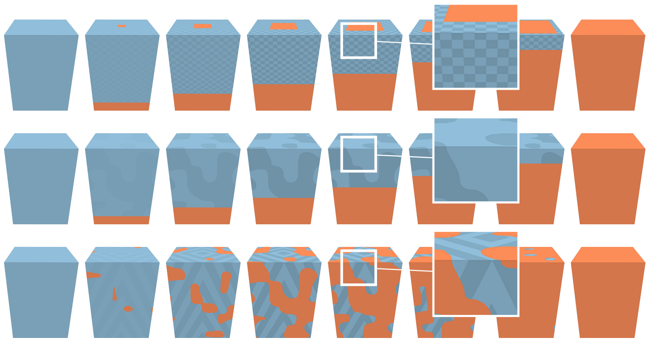


Figure 6: Combination of two procedural patterns. The dominant pattern is supported by a secondary pattern placed only on the parts representing either the former (as can be seen here) or the upcoming value.

Candidates for Secondary Use. Technically, the secondary pattern is not limited by the choice of the main pattern. However, we suggest using a different, distinct pattern w.r.t. to the main one, ideally one that has a uniform distribution of colors. Both approaches ensure that the pattern is (1) distinguishable from the main pattern and (2) visible without regard to the current transition state.

Parameterization of Secondary Pattern. The secondary pattern itself is derived for a static transition value of $t = 0.5$ and applied using blending of the base color and a slightly darker variant. This way, the pattern occupies half the area of the applied surface. Regarding the choice of the applied surface, we propose to use the surface that encodes the former color to superimpose the second pattern. When using the former color, the advantage is that the secondary pattern can be used until the end of the transition, as the surface of former color will vanish during the transition. In addition, the secondary pattern must not be faded in completely directly at the beginning of the transition. Otherwise, if an animation consists of several transitions, the secondary pattern would appear abruptly when two transitions are changed. Therefore, the secondary pattern is faded in gradually with the help of an easing.

2.3 Transitions and Animations

So far, we considered isolated transitions from one color to another for a single node. In order to display multiple changes over multiple snapshots of data, we consider the use of animations.

Transition. A transition describes a change from a nodes state or mapped value to another one. Thereby, a transition will not encode time or duration but only progress. The required parameter for such a transition is its progress control $t \in [0, 1]$ where 0 is used at the start and 1 at the end of a transition.

Animation. Animation is the continuous progression over multiple states by means of transitions. More important, it is linked to the concept of time and duration. Note that the transitions can be enhanced by easing functions which is not discussed in this context [11]. The animation between two points in time, e.g., two subsequent revisions for software system data, can range from running all transitions for every node and every visual variable in

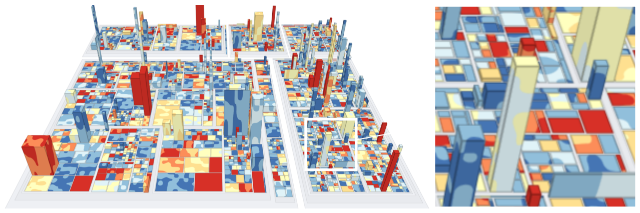


Figure 7: Example of an extreme case with all color transitions of all nodes beeing shown at once with $t = 0.5$.

parallel as well as in strict sequence. The former approach is strictly limited by the number of concurrent changes (Figure 7).

3 DISCUSSION

Our prototype extends an existing, WebGL-based rendering system for large-scale 2D and 2.5D treemaps. The implementation of the patterns itself was straightforward and imposed no significant impact on rendering performance (runs on mobile devices for large treemaps). For the animation in general, we had to redesign large parts of our visualization and rendering implementation though. In the remainder of this section we briefly discuss some animation aspects and assess the applicability of our patterns.

3.1 Animation Controller

We composed the handling of animations and transitions to a system that manages all temporal animations, controlling the time-dependent transition values as well as the individual, per-node transitions states (Figure 1). Thereby, our prototype allows for independent, simultaneous transitions per-node, for every mapped data attribute. The procedural patterns are implemented as temporal visual variables, mapping a 3-tuple (a value per state and transition progress) to color instead of single value. An animation of weight and height mappings are implemented similarly. The latter allows for ordering and masking of transitions based on (1) transition type, (2) arbitrary node associated values, (3) meta data, or (4) enumeration and more. With this starting point, we have begun to identify meaningful applications with this design space.

3.2 Qualitative Assessment

We assessed the temporal visual variables, using every pattern individually and in combination, and applied them for weight, height, and color change emphasis on small and large real-world data sets. For the scope of this work, we derive the following characteristics:

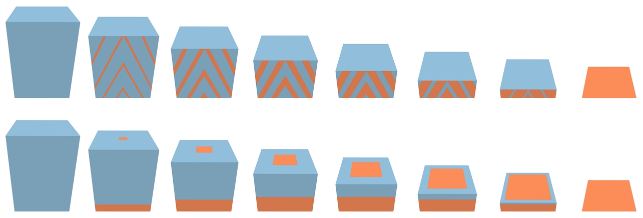


Figure 8: Example of two patterns used for a transition of simultaneous increase in color and decrease in height values.

Table 1: Overview of the patterns and their characteristics. Notation: ● supported | ○ partial support | – unsupported.

Procedural Pattern	Δ_{DirAnim}	$\Delta_{\text{DirChange}}$	Δ_{RatioLat}	Δ_{RatioTop}	$\Delta_{\text{IndHeight}}$	$\Delta_{\text{IndWeight}}$	Δ_{TreeSize}
Pillar	●	–	○	–	–	●	●
Pyramid	●	–	●	●	–	○	●
Dithering	–	–	●	●	●	●	●
Arrows	–	●	○	–	●	●	●
Arrows (full)	–	●	●	●	●	●	●
Noise	–	–	–	●	●	●	–
Squares	–	–	●	●	●	●	–

Δ_{DirAnim}	The direction of animation can be read unambiguously while the animation is paused in a transition.
$\Delta_{\text{DirChange}}$	The direction of change in value can be read unambiguously while the animation is paused.
Δ_{RatioLat}	The ratio of different colors on the lateral faces matches a transition's progress.
Δ_{RatioTop}	The ratio of different colors on the top face matches a transition's progress (top view or 2D treemap).
$\Delta_{\text{IndHeight}}$	The mapping of change difference and change direction does not compromise the height mapping.
$\Delta_{\text{IndWeight}}$	The mapping of change difference and change direction does not compromise the weight mapping.
Δ_{TreeSize}	The pattern can be easily adjusted for large or very large treemaps.

We assessed all proposed patterns on real data sets by their degree to satisfy the characteristics (Table 1). We conclude that no pattern satisfies all desired characteristics. However, we want to highlight the arrows (full) pattern that supports most of the characteristics.

4 CONCLUSIONS

The proposed system supports display of changes in all of treemap item areas, heights using geometry displacement, and colors using animated procedural textures. This is relevant for the exploration of time-varying data such as software data. As a special requirement we introduced the readability for both a static image and a dynamic animated transition. For specific templates on how to transition color values in 3D we proposed the seven variants (1) *pillar*, (2) *pyramid*, (3) *arrows*, (4) *arrows (full)*, (5) *noise*, (6) *dithering*, and (7) *squares*. We discussed the different characteristics, especially direction of change and direction of animation, strengths, and weaknesses of each template but cannot draw a conclusion on a default template at this point. However, we found that using the time control helps to understand the changes of individual nodes.

For future work, we plan to perform quantitative evaluations by means of comprehensive user studies and extended case studies on large software systems and long time spans.

ACKNOWLEDGMENTS

This work has been supported by the German Federal Ministry of Education and Research (BMBF) through grant 01IS19006 ("KI-Labor ITSE"). Further, this work is part of the "Software-DNA" project, which is funded by the European Regional Development Fund (ERDF or EFRE in German) and the State of Brandenburg (ILB), and the "TASAM" project, which is funded by the German Federal Ministry for Economic Affairs and Energy (BMWi, ZIM).

REFERENCES

[1] T. Bladh, D. A. Carr, and M. Kljun. 2005. The effect of animated transitions on user navigation in 3D tree-maps. In *Proc. Ninth International Conference on Information Visualisation (IV'05)*. 297–305. https://doi.org/10.1109/IV.2005.122

[2] Yi Chen, Xiaomin Du, and Xiaoru Yuan. 2017. Ordered small multiple treemaps for visualizing time-varying hierarchical pesticide residue data. *Springer The Visual Computer* 33, 6 (2017), 1073–1084. https://doi.org/10.1007/s00371-017-1373-x

[3] Alexandre Henrique Ichihara Pires, Rodrigo Santos do Amor Divino Lima, Carlos Gustavo Resque dos Santos, Bianchi Serique Meiguins, and Anderson Gregório Marques Soares. 2020. A summarization glyph for sets of unreadable visual items in treemaps. In *2020 24th International Conference Information Visualisation (IV)*. 242–247. https://doi.org/10.1109/IV51561.2020.00047

[4] G. Kindlmann and C. Scheidegger. 2014. An Algebraic Process for Visualization Design. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2181–2190. https://doi.org/10.1109/TVCG.2014.2346325

[5] G. Langelier, Houari Sahraoui, and P. Poulin. 2008. Exploring the evolution of software quality with animated visualization. In *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing*. 13–20. https://doi.org/10.1109/VLHCC.2008.4639052

[6] Daniel Limberger, Trapp Matthias, and Jürgen Döllner. 2019. In-situ Comparison for 2.5D Treemaps. In *Proc. 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 3: IVAPP*. INSTICC, SciTePress, 314–321. https://doi.org/10.5220/0007576203140321

[7] Daniel Limberger, Willy Scheibel, Matthias Trapp, and Jürgen Döllner. 2019. Advanced Visual Metaphors and Techniques for Software Maps. In *Proc. 12th International Symposium on Visual Information Communication and Interaction (VINCI '19)*. ACM, 11:1–8. https://doi.org/10.1145/3231622.3231638

[8] Ken Perlin. 2001. Noise Hardware. In *Real-Time Shading SIGGRAPH Course Notes (2001)*, Marc Olano (Ed.). Chapter 9.

[9] Willy Scheibel, Matthias Trapp, and Jürgen Döllner. 2016. Interactive Revision Exploration using Small Multiples of Software Maps. In *Proc. 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 2: IVAPP, (VISIGRAPP 2016)*. INSTICC, SciTePress, 131–138. https://doi.org/10.5220/0005694401310138

[10] Sidharth Thakur and Theresa-Marie Rhyne. 2009. Data Vases: 2D and 3D Plots for Visualizing Multiple Time Series. In *Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009. Proceedings, Part II*. Springer Berlin Heidelberg, 929–938. https://doi.org/10.1007/978-3-642-10520-3_89

[11] Frank Thomas and Ollie Johnston. 1981. *Disney Animation: The Illusion of Life*. Abbeville Press.

[12] Y. Tu and H. W. Shen. 2007. Visualizing Changes of Hierarchical Data using Treemaps. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1286–1293. https://doi.org/10.1109/TVCG.2007.70529

[13] R. A. Ulichney. 1988. Dithering with blue noise. *Proc. IEEE* 76, 1 (1988), 56–79.

[14] Patricio Gonzalez Vivo and Jen Lowe. 2015. *The Book of Shaders*. https://thebookofshaders.com/

[15] Hannes Würfel, Matthias Trapp, Daniel Limberger, and Jürgen Döllner. 2015. Natural Phenomena as Metaphors for Visualization of Trend Data in Interactive Software Maps. In *Proc. Computer Graphics and Visual Computing (CGVC)*. 69–76. https://doi.org/10.2312/cgvc.20151246