Spatial Text Editor Caret Navigation in Code Containing Text and Visual Editors

A Caret for Your Thoughts: Adapting Caret (𝙸) Navigation to Visual Editors
What if text editors supported drawing a decision tree inside of code instead of writing a sequence of if-statements; or typing nicely formatted math equations; or embedding editable graph diagrams; or embedding image editors? In general, what if code editors could embed visual editors in text? I've been exploring this question through building Polytope, a code editor prototype that combines text and visual editors.

In a text editor, we navigate through code by moving a caret (𝙸) using our keyboard and mouse. In Polytope, navigation has to work between both text and visuals. Can we adapt caret navigation to visual editors, to create a bridge between navigating text and visuals? In particular, can we translate index-based caret navigation into spatially-based caret navigation for visuals?

This talk is a follow up to my PX/22 talk: let chart = ⊥; let song = ♩; // Embedding Visual Languages in Code.

Spatial Text Editor Caret Navigation in Code Containing Text and Visual Editors

We all use a text editor to write source code.

What if our text editors supported drawing a decision tree inside of a code file instead of writing a sequence of if-statements; or supported writing nicely formatted math equations; or embedding graph diagrams; or embedding image editors? In general, what if our text editors supported writing programs using text inline with these sorts of visual editors?

I created a prototype of a text editor, called Polytope, that allows me to write code containing visual editors. In Polytope, I can choose to use a visual editor in part of the code instead of text. Each editor has custom editing controls. For example, I can write `const graph = `, insert a graph diagram editor, click to create nodes in the editor, and finally drag my mouse between nodes in the network to create connections. In order to save or evaluate my code, Polytope turns editors into text. When loading my code from a file, Polytope turns the appropriate text back into editors.

In a text editor, we navigate through code by moving a caret (𝙸) using our keyboard and mouse. In text, navigation can already be cumbersome. In Polytope, code contains a mixture of both text and visuals, which involves navigating between both text and visuals. We can adapt caret-based navigation to visual editors, to create a bridge between navigating text and visuals. so it is even more important to consider ease of navigation through code. Caret-based navigation can be generalized to work for visual editors to make navigation seamless between

text and visual editors. Specifically, caret-based navigation can be adapted for visual editors by translating caret-based navigation specification (which is traditionally index-based) into a spatially-based specification.

If you're interested in visual programming, combining text and visual programming, or graphics programming, I hope you will enjoy this talk. Thank you.

Do you ever find it easier to read math equations than math in code? Rapidly switch between an image editor and an SVG editor?

- how can I explain what I mean by notation?
- how can I explain what keyboard navigation in notations means?
- what does seamless mean?
- What is topology?

Have you ever wanted to write code containing a nicer notation, such as a equation Σ, sheet music ♩, or a decision tree Ψ? What if you could edit these notations as if they were literals in your code? To explore these questions I built Polytope, a text editor that enables embedding visual languages in code.

Is it possible to make navigating various notations in a text editor feel seamless? In particular, can we make it so that keyboard caret navigation is consistent in-between notations? The answer is yes! We can make navigating various notations in a text editor feel seamless using spatial navigation with some text-editor-y constraints.

Let's explore how to visualize the graph underlying text-editor navigation, the properties that make a text-editor feel like a text-editor, and how we can approach building a bridge between text language editing and visual language editing.

This talk is a follow up to my PX/22 talk: let chart = ⊥; let song = ♩; // Embedding Visual Languages in Code.

Text Editor Topology: generalizing text editors to accommodate visual languages

Have you ever wanted to write code containing a nicer notation, such as a equation Σ, sheet music ♩, or a decision tree Ψ? What if you could edit these notations as if they were literals in your code? To explore these questions I built Polytope, a text editor that enables embedding visual languages in code.

Is it possible to make navigating various notations in a text editor feel seamless? In particular, can we make it so that keyboard caret navigation is consistent in-between notations? The answer is yes! We can make navigating various notations in a text editor feel seamless using spatial navigation with some text-editor-y constraints.

Let's explore how to visualize the graph underlying text-editor navigation, the properties that make a text-editor feel like a text-editor, and how we can approach building a bridge between text language editing and visual language editing.

This talk is a follow up to my PX/22 talk: [let chart = ⊥ ; let song = ♪; // Embedding Visual Languages in Code]().

- Alt title: Navigating visual languages like text languages
- Text-Editor Micro-Interactions

- Once a notation is embedded in text, how can it be navigated? Is it possible to make the navigation feel seamless in the context of a text editor?
- Visualize caret position graph. What properties does it have? It is linear. Let's visualize the properties and talk about them. What if we moved the caret positions all around and stuff? Can we get the same properties? Let's try. Yup, we can get some nice properties!
- further work: editing, selections, copy-paste, history, proving in Agda, projections, multiplayer, editor editors,...
- A follow up to my 2022 talk: https://2022.programming-conference.org/details/px-2022-papers/7/let-chart-let-song-Embedding-Visual-Languages-in-Code