

# ABOUT THIS TYPESETTING EDITOR

This typesetting editor is an application I have been developing for over half a year. The codebase was written from scratch in Python 3, and uses its own UI toolkit which I also wrote. Although still in alpha stages, the codebase can be synced from <https://github.com/kelvin13/shifty-octocat>, where it will work out of the box on any Linux installation.

$$\int_{13}^{22+\frac{e}{2}} \int_{13}^{22+\frac{e}{2}} - \frac{abc + \frac{e}{n}}{22+\frac{e}{2}} \frac{abc + \frac{e}{n}}{\int_{13}^{22+\frac{e}{2}} \frac{e}{n}} \frac{abc + \frac{e}{n}}{n} \frac{abc + \frac{e}{n}}{n}$$

$a - \int_{13}$

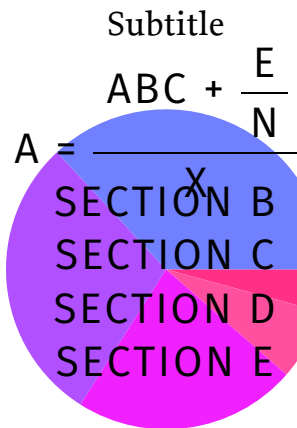
The end goal of the project is to include all the tools and parameters handled by “serious” typesetters, and allow editing of them

in the least verbose manner. Hence, Vedit offers granular control,  
with a strong emphasis on **additive styling**.  $\frac{a}{x}$

—*Kelvin Ma*

## WHAT IS ADDITIVE STYLING?

### PIE CHART EXAMPLE



Additive styling is the idea that multiple “building block” text

styles can be defined and combined to create the visual styles seen on the page. For exam

CELL IN- TEGRA- TION	CELL 2 (COLUMN SPAN = 3)		
cell 3 Nor- ton inter- national- ization	cell 4	cell 5	cell 6
cell 3 (row 2)	cell 7	cell 8	cell 9

ple, an *<Italic>* class may be defined which simply declares an italic font file, while a <sup>Superscript</sup>< > class might define a reduced font size and raised position. The superscript and italic classes can then be stacked to infer a <sup>*Superscript-Italic*</sup>< > class with the at-

tributes of both.

Of course, in real typesetting, superscripts require a slightly heavier fontweight, which is where **explicit styling** is needed. An italic and a bold class both define a font file, so a **<Bold-Italic>** class must be defined with the specialized bold-italic font.

Additive styling is nevertheless very powerful. *Classes can overlap each other on the page, allowing **mixing effects**, such as in this sentence where a pair of italic tags mixes with a pair of bold tags over the word “interesting”.* Additive styling even allows the use of negative styles, and multistyles. *This lets us do things such as **superitalics** or **superbolds** in our text.* Negative styles are work too, though there is little practical use for them.

Additive styling to my knowledge is unique to Vedit, though CSS can be hacked to achieve a significant subset of its behavior. It is distinct from an **inheritance model** because additive styling is flat rather than nested, relying on an ordered sequence of classes which can be turned on and off to effect certain elements on the

page. Unlike the **flat styling model**, additive styles mix and combine with one another, eliminating redundant statements, and unlike the **direct styling model**, it preserves the order of editable classes.

Paragraphs too, have additive styling, which functions in the same way as text styling, except with attributes like leading, hyphenation, margins, and indentation.

## PARAGRAPH CONTROL

Instead of “single” or “double” spacing, Vedit lets the writer specify the literal point size of the **leading**, or line height. This helps you predict the amount of space that text will take up, and helps you avoid text alignment headaches, since the spacing is always constant and measurable.

**Indentation** is powerful in Vedit. The lines over which it takes effect can be explicitly enumerated (in a relative count from the start of the paragraph), and it can be set in terms

of the character positions of  
the first line.

—> This is very use-  
ful for creat-  
ing  
**han**

*interesting* con-

structs, such as this

gray filter, which is made up of

a paragraph class containing a single

text class which defines just a gray color, all

overlayed over our standard body text style.

## SUPERSCRIPTS AND SUBSCRIPTS

**gin**  
**g**  
**inde**

Integral test:  $\int_{13}^{22 + \frac{e}{2}} f(x) dx.$

## TEXT CHANNELS

**nts.**

Para-

graph styles

even add their

**constituent text styles,**

which allows for *very*

One last thing I might want to mention. Text columns

are completely flexible, and they can be bent to

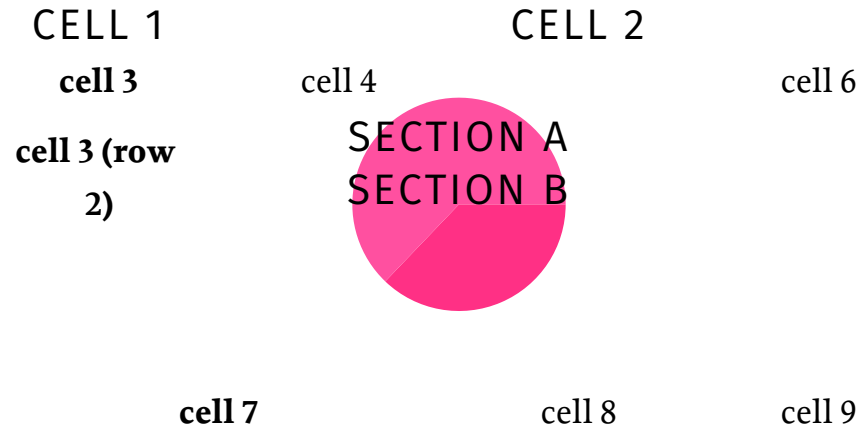
make any text wrapping you would ever

need. Even *negative* text wrap-

ping works.

Flexible text

channels even work with Vedit's tables, which calculate their cell width based on the width of the text channel at that point.



Of course, square tables are much easier to read.