Slider Velocity Crash Course

Evening

Contents

Part I
Preface

1 First Look

1.1 What does SV actually stand for

Slider Velocity, or more representative of mania, Scroll Velocity. But never Speed Velocity.

1.2 What you need to know before learning starting to read this document

You just need basic knowledge of Mathematics and preferably understanding of the "Distance, Time and Speed" concept.

1.3 Why SVs are important in mapping/modding

Modding-wise, you're most likely going to face a challenge of modding an SV-oriented map, if you understand how to use it, you'll understand how to fix it.

Mapping-wise, most mappers take it as a choice to use it or not, some seasoned mappers probably haven't touched it. The only part where you need SVs is when you start to encounter multi-BPM charts, you'll definitely need to utilize normalization only achieve-able in SVs.

Part II
Introduction

This document will mainly touch on basics of SV creation, most of the advanced work on SVs should be done by you. This will not touch on anything advanced, such as:

- Complex SV creations
- BPM Mapping
- Automated Creation of TimingPoints
- p-Notes / Fake Notes

I'll try to be as unbiased as possible when writing this document, please take this as a point of reference rather than a rule of thumb, the limitations much further than what this document will cover.

2 Starting Out

2.1 Adding your First SVs

- Ctrl+Shift+P adds an SV Line
- Ctrl+P adds a BPM Line

Note for **new mappers**:
*It is possible to add multiple SV/BPM lines at one single time stamp. For most cases, SVs on top of SVs **OR** BPMs on top of BPMs will generate undefined behavior, so try to avoid it.*
*However, SVs on top of BPMs may be used to **Normalize**, in which will be a future covered topic.*

2.2 Defining your SV Values

The value directly **multiplies into** the scroll speed the player is currently running on, so **2.0** scrolls twice as fast as **1.0**.

2.2.1 Boundaries

osu!mania only supports $0.1 \longleftrightarrow 10.0$.
Contrary to being able to use $0.01 \longleftrightarrow 0.10$, there is no effect on the map.

2.3 BPM Effects

If you have encountered a multi-BPM map, you'll notice that BPM lines will affect scroll speed too. It works similar to SVs, where **200 BPM** sections will scroll twice as fast as **100 BPM** sections.

2.4 Always consider Slider Velocities over BPM Lines

There are 2 ways to manipulate scroll speed, through BPM and SV. You can take it as a rule of thumb to **NEVER** use BPM to change scroll speed unless you know what you're doing! I'll list down the pros and cons below.
Take note that the context is to only use BPM for scroll speed manipulation

2.4.1 Pros of BPM usage

- Very wide limits ($0 \longleftrightarrow inf$)
- Can generate/affect measure lines
- Arguably easier to calculate when used in Multi-BPM maps

2.4.2 Cons of BPM usage

- Not intuitive on creation of extreme values
- Inaccuracies when rounding to 3 decimal places *eg.* ~ 33.333
- Due to osu!mania being reliant on snap mapping, it will be not intuitive to map with an altered BPM when creating effects.
- BPM Lines will lie on integer offsets, hence with non-integer beat lengths, the original BPM offset will start to sway.
- Extreme BPM lines can crash the editor
- And many more...

To summarize, do not use BPM Lines unless you know what you are doing.
I will not be covering BPM scroll speed manipulation in this document.

Part III
Normalization

3 Normalizing a Multi-BPM Map

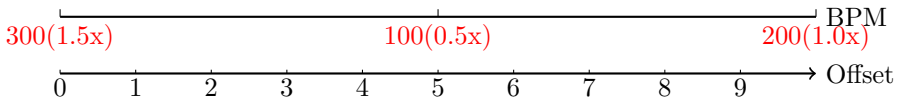
Even if you don't consider **ever** mapping/modding a multi-BPM map, you will still find this concept useful. So unless you fully understand **Normalizing**, do not skip this chapter!

3.1 The idea of "Normalization"

Normalization in this context means adding SV lines over BPM lines in order to force the map to run at a **constant scroll speed**, that is, **1.0x**.

3.2 Normalizing a simple Beatmap

3.2.1 A 200 Reference BPM Map



Offset	BPM	$Scroll_{BPM}$
0	300	1.5x
5	100	0.5x
10	200	-

This is one of the more common cases of multi-BPM that you'll come across, where the song itself will change BPMs in a simplistic way.
Take note of the bracketed value beside each BPM (eg. 300**(1.5x)**), this indicates the scroll speed modifier if there aren't any SVs.

3.2.2 Calculating the Normalization SVs

The aim is to calculate the value to counteract scroll speed modifier from BPM lines.

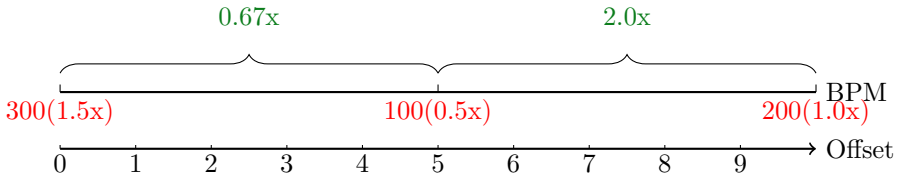
In order to counter the 1.5x Scroll Speed and adjust it to 1.0x. We add a Normalizing SV that is:

$$SV_n = \frac{1.0}{1.5} \approx 0.67(2d.p.)$$

We counter the 0.5x Scroll Speed and adjust it to 1.0x. We add a Normalizing SV that is:

$$SV_n = \frac{1.0}{0.5} = 2.0$$

3.2.3 A 200 Reference BPM Map (Normalized)



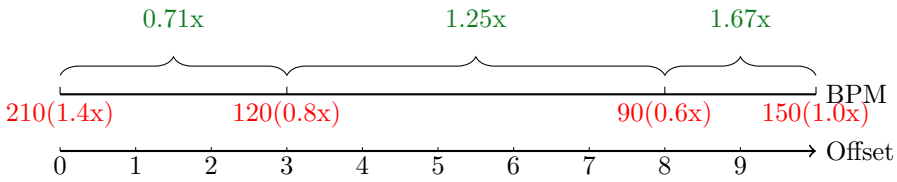
The **Final** Scroll Speed is defined by the *Scroll Speed Modifier of the BPM*, multiplied by *Scroll Speed Modifier of SV*, which is just the value itself. Hence:

Offset	BPM	$Scroll_{BPM}$	SV	$Scroll_{Final}$
0	300	1.5x	0.67x	~1.0x
5	100	0.5x	2.0x	1.0x
10	200	-	-	-

$$Scroll_{BPM} * SV = Scroll_{Final} = 1.0$$

I'll illustrate another more complex example.

3.2.4 A 150 Reference BPM Map (Normalized)



$$SV_{n1} = \frac{150}{210} \approx 0.71$$

$$SV_{n2} = \frac{150}{120} = 1.25$$

$$SV_{n3} = \frac{150}{90} \approx 1.67$$

Offset	BPM	$Scroll_{BPM}$	SV	$Scroll_{Final}$
0	210	1.4x	0.71x	~1.0x
3	120	0.8x	1.25x	1.0x
8	90	0.6x	1.67x	~1.0x
10	150	-	-	-

3.2.5 In General

$$NormalizeSV = \frac{ReferenceBPM}{CurrentBPM}$$

3.3 Reference BPM

You have seen this value a lot in this topic, but *how* to do you get this value? The reference BPM is basically the value **in brackets** shown above. In this case, simply **156**.

3.3.1 Calculation of Reference BPM

The Reference BPM is reliant on the longest duration mapped on a BPM. Hence, *there is a possibility* that the Reference BPM will change, and you'll have to calculate all over again.

3.3.2 Avoiding multiple Normalizations

One way to do so is to litter the whole map with notes, **with intended breaks**, so that the game can process which reference BPM would be used, once you finished the whole chart.

The other would be to calculate manually, which is not recommended for complex cases.

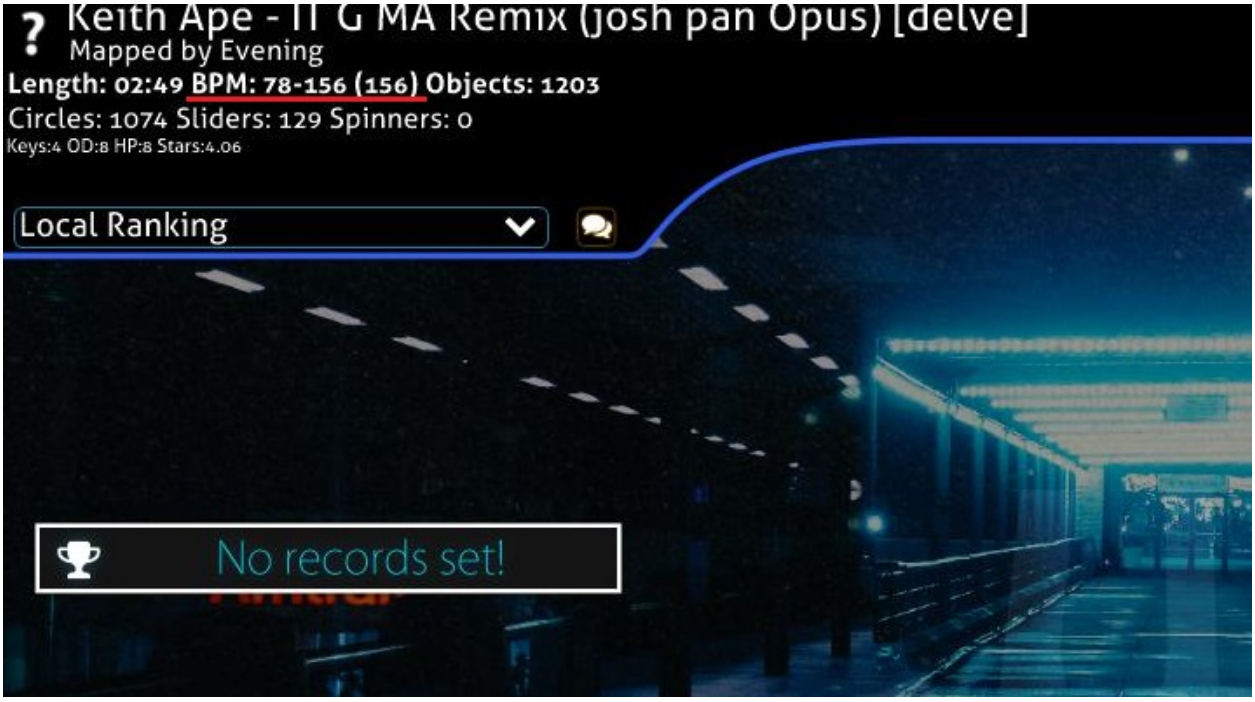


Figure 1: Reference BPM Example

3.4 Tools for Normalization

While you can just use these tools, it is important for you to understand this topic as a foundation to the next topics. **The tools can be found in the Annex**. I'd recommend using Agka's SV tool over mine as it's more convenient.

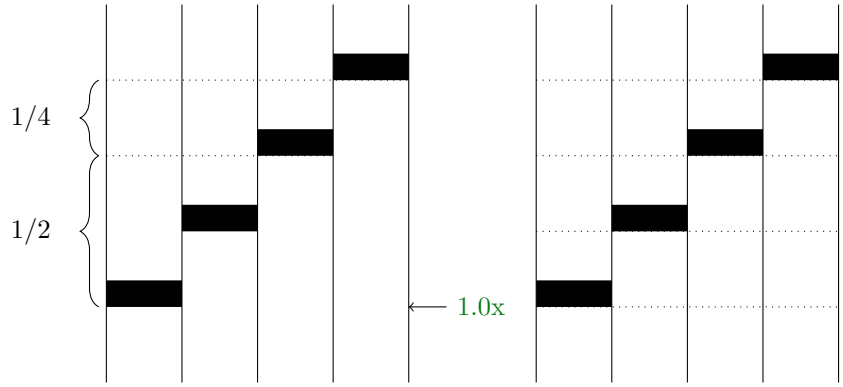
Part IV
Simple Effects

If you've read the previous topic, you should be able to understand this topic better, if you haven't, that's fine, I'll still guide you through from square one.
Before we go into putting multiple SVs together, we have to understand the effects of SVs, down to the core. We'll start with something that is not commonly talked about: **Visual Spacing**

4 Visual Spacing

Visual Spacing is the space in between notes that players will see, this gives the indication on if the note is a certain snap. Eg. 1/4 notes will always have **half** the spacing of 1/2 since it's **twice as fast**.
Consider this scenario. Visually, all of these diagrams look like they are 1/4 snaps

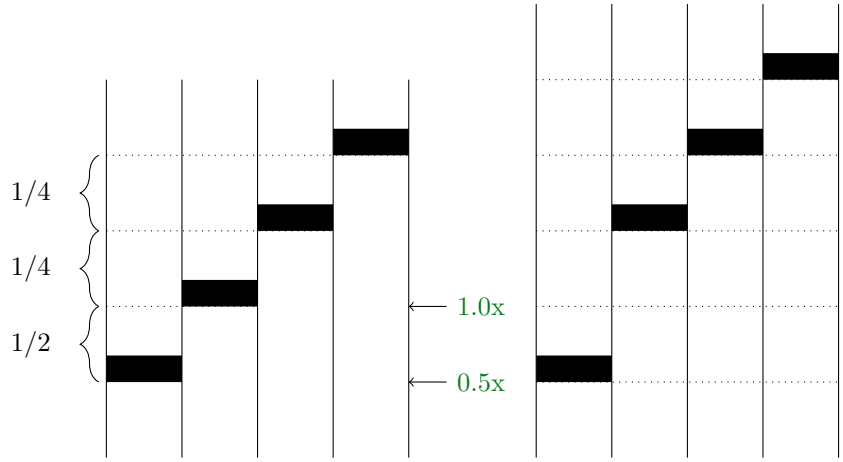
4.1 Visual Spacing Scenario



Gameplay View (left) vs. **Editor View** (right)
Each dotted line in Editor View represents 1/4

It's obvious how that works, but watch what happens when you **add SVs and move notes around** on the next scenario.

4.1.1 Uneven Visual Spacing Scenario 1



Gameplay View (left) vs. **Editor View** (right)
Each dotted line in Editor View represents 1/4

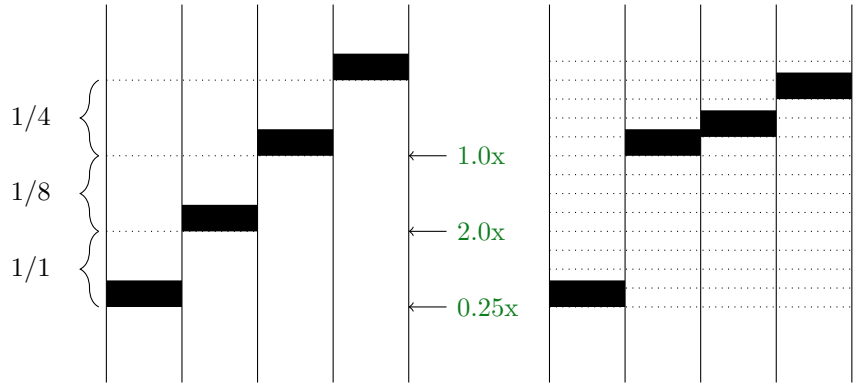
4.2 Uneven Visual Spacing

With regards to what an SV does, a 0.5x SV will make **1/2 spacing** visually look like **1/4 spacing** but it will STILL play as if it was on **1/2 spacing**. This is an *uneven* SV Effect because it distorts the **Visual Spacing**.

4.3 Downsides of Uneven Visual Spacing

With **unexpected** uneven visual spacing/SV, players will be **expecting a certain snap**, but instead, is **greeted with another snap**.
Referring to the previous diagram, we can see that the player will be **expecting a 1/4 snap** for all 4 notes. However, since a **0.5x** SV was implemented, **the first 2 notes will play as if it was 1/2 apart** Here's another example for easier digestion

4.3.1 Uneven Visual Spacing Scenario 2



Gameplay View (left) vs. **Editor View (Scale: 0.5)** (right)
Each dotted line in Editor View represents 1/8

Now you understand the effects of SV on notes, we can go onto how to make SVs that create **Even Visual Spacing**, in the following topic.

5 Act, Counteract and Closure

For one of the simplest and most common effects, most mappers goes through 3 steps of creation

- Act: The main effect you want to create
- Counteract: The effect that *evens out the Visual Spacing*
- Closure: This is usually to normalize the SV back to 1.0, as to "close" the sequence

Let's take a look at each of these in detail and what they do.

5.0.1 The Act

As explained, this is the **main source** of the effect. What you want to achieve in the effect should be done here.
Take for example:

- **An effect that noticeably jumps forward once a note is hit**
  We should expect a *reasonably high value SV* to achieve this effect.
  For simplicity, we will only use up to 1.9, I will explain why in the **Half-half Concept** topic.
- **An weak effect that will barely affect gameplay, but enough for players to notice once a note is hit**
  We should expect a *relatively weak SV* as compared to the above example.
  We can consider anything in the range of $1.2 \longleftrightarrow 1.5$ where is tame, but not too close to 1.0.

5.0.2 The Counteract

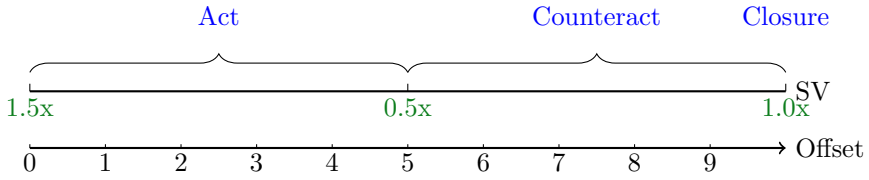
As you'll expect, you will end up with **Uneven Visual Spacing** if you just stop here, so this one mainly targets that issue and corrects it. We'll go into detail how to do this later.

5.0.3 The Closure

The closure is the SV that closes the SV sequence, returning everything to normal. This is so that the rest of the chart isn't affected by **Act or Counteract SV**.

5.1 Breakdown of Effects

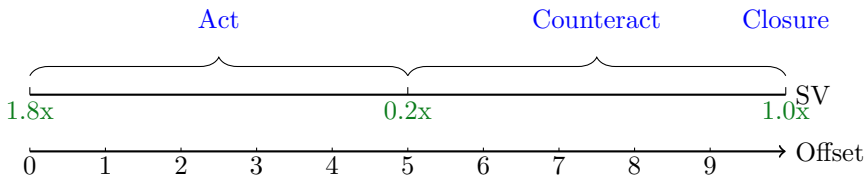
5.1.1 Breakdown of a Simple Effect [1]



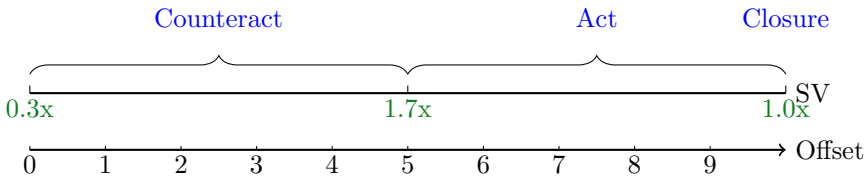
This is one of the most common and simpler effects to show. This produces a "jump" effect, where it speeds up, then comes to stop. The "jump" is mainly shown by **1.5x**, hence it's the Act. The "fix" is reliant on the value of your Act, it is calculated to be **0.5x**.

Last but not least, the closure, the scroll speed multiplier that you want the rest of the chart to go at.
Here's two more examples on how this works

5.1.2 Breakdown of a Simple Effect [2]



5.1.3 Breakdown of a Simple Effect [3]



5.2 The Half-Half Rule

This rule works **if and only if** *your second SV is halfway in between the first and third.*

Notice how all of the effects have certain values that work, here's short list of valid SVs that works in those situations, see what you can deduce from this

SV_{Act}	$SV_{C.Act}$	$SV_{Closure}$	$SV_{Average}$
1.9x	0.1x	1.0x	1.0x
1.5x	0.5x	1.0x	1.0x
1.3x	0.7x	1.0x	1.0x
1.1x	0.9x	1.0x	1.0x
0.9x	1.1x	1.0x	1.0x
0.7x	1.3x	1.0x	1.0x
0.5x	1.5x	1.0x	1.0x
0.1x	1.9x	1.0x	1.0x

We can say that...

$$\frac{SV_{Act} + SV_{C.Act}}{2} = SV_{Closure} = SV_{Average}$$

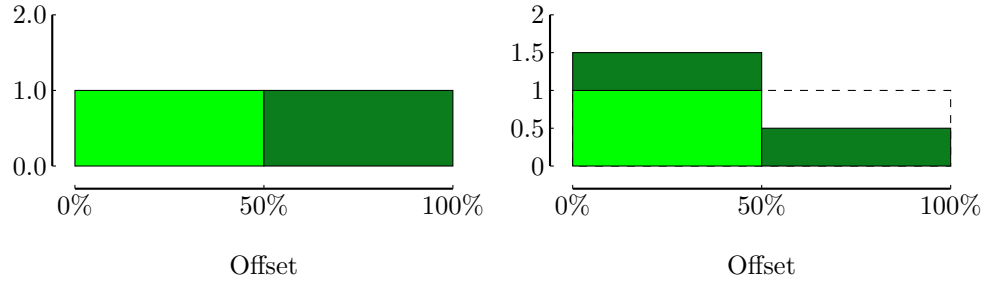
OR
If your $SV_{Average} = 1.0x$, then...

$$SV_{Act} + SV_{C.Act} = 2.0$$

This is the **Half-Half Rule**, all SVs created with this **will average to 1.0**.

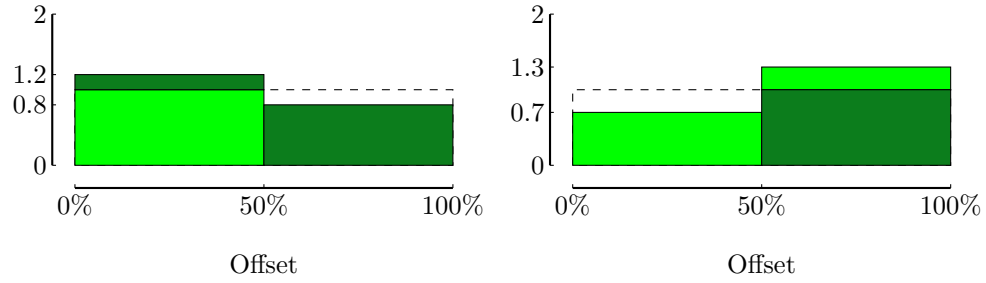
5.2.1 How it Works

Avoiding the more complicated mathematics, we can explain this with some diagrams.



This is a simplified diagram for a **Half-Half Rule** situation, all SVs are represented in the boxes. When we **displace** 0.5x SV we can easily shift it to the other SV so it still has **the same area**. In other words, if it has **the same area**, it will always be **Even Visual Spacing**.

Let's take a lot at a few more examples:



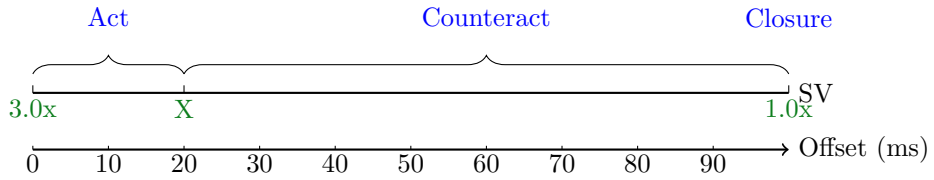
This is simple mathematics, we actually don't need diagrams. However, this is essential in helping you understand the more dynamic cases, where you can't use the **Half-Half Rule**.

Part V
Advanced Effects

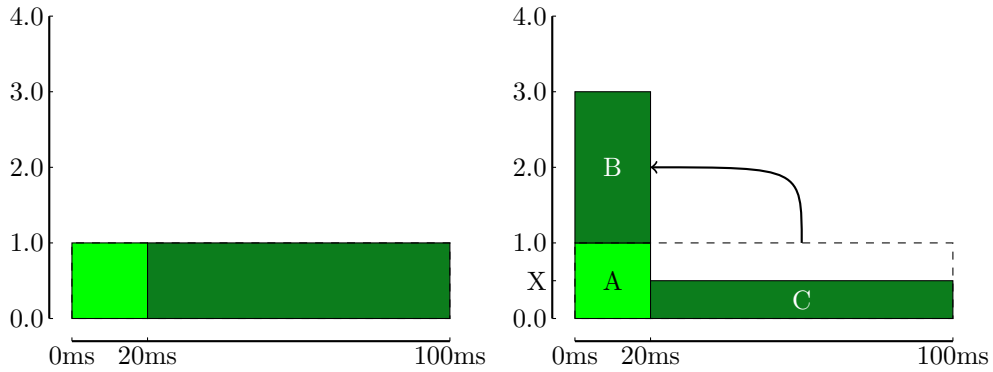
This will be the final chapter in the document. So if you've understood everything so far, you're well on your way to mastering it already.

6 Visualizing Everything

6.1 Breakdown of an Advanced Effect [1]



How would we get the value required to create **Even Visual Spacing**? The simple answer is to refer back to the diagrams we drew previously and work from there.



No SV Reference *(left)* vs. SV Reference *(right)*

One way we can deduce the value there is through **calculation of Area**.

Before we delve into the calculation, we need to first know what is the shaded area called. I personally coin it as **Distance** because it defines how far the notes will travel, and **Distance** defined as:

$$SV * Offset = Distance$$

Important Rule of Thumb:

SV Sequences with equal Distances will have Even Visual Spacing with respect to each other

6.1.1 Calculation with Geometry

This is the simpler but longer way of calculating the unknown. If you prefer using **equations** you can skip to the following section.

First and foremost, we need to understand that in both diagrams, they have the same shaded area (Distance), so if we can calculate the *shifted area* (indicated by the arrow), we can calculate X.

We first calculate the shifted area as shown:

$$Area_B = SV * Offset = 2.0 * 20 = 40$$

$$Area_B + Area_C = SV * Offset = 1.0 * 80 = 80$$

$$Area_C = (Area_B + Area_C) - Area_B = 80 - 40 = 40$$

$$Area_C = SV * Offset = X * 80 = 40$$

$$X = 0.5$$

Note: There are some steps that you can avoid with different approaches.

6.1.2 Calculation with Equations

This is the harder but shorter way of calculating the unknown. If you prefer using **Geometry** read the previous section.

Looking at the **Left Diagram**, we can calculate **Distance**:

$$Distance = SV_{Total} * Offset_{Total} = 1.0 * 100 = 100$$

Note: We can usually skip this step and assume $Distance = Offset$ because most of the time we use 1.0x as the SV_{Total}, and we know anything multiplied by 1.0 is the same.
We know that the total of the colored area (Distance) never changes because Distance remains the same so that Visual Spacing remains even. We can then calculate X.

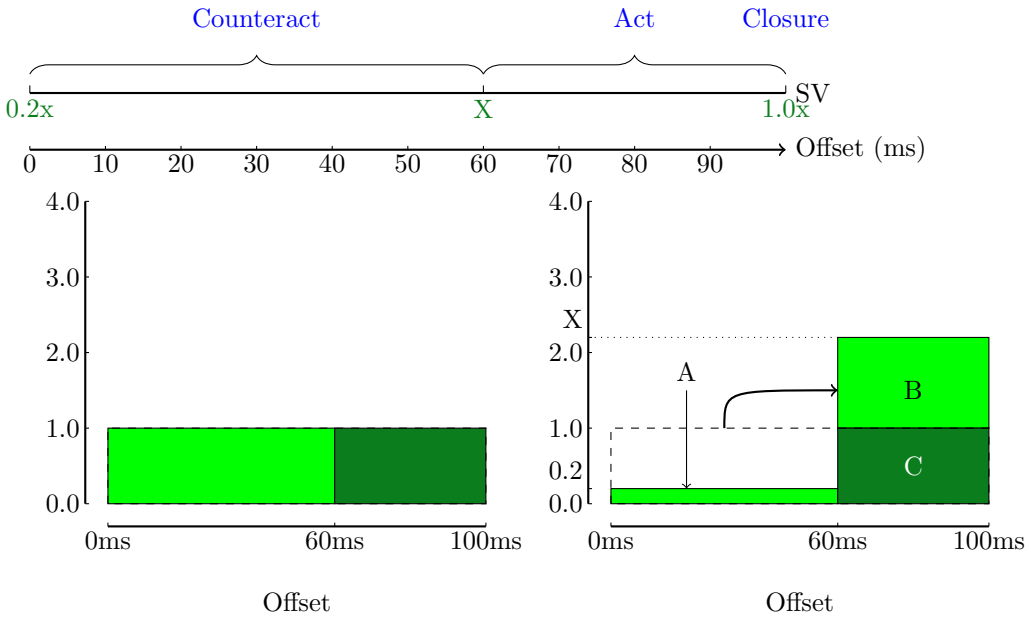
$$SV_1 * Offset_1 + SV_2 * Offset_2 = Distance = 100$$

$$3.0 * 20 + X * 80 = 100$$

$$X = 0.5$$

These are the 2 ways you can calculate unknown SVs. Let's take a look at more examples:

6.2 Breakdown of an Advanced Effect [2]



No SV Reference *(left)* vs. SV Reference *(right)*

6.2.1 Calculation with Geometry

$$Area_A = SV * Offset = 0.2 * 60 = 12$$

$$Area_A + Area_B = SV * Offset = 1.0 * 60 = 60$$

$$Area_B = (Area_A + Area_B) - Area_A = 60 - 12 = 48$$

$$Area_B = SV * Offset = (X - 1.0) * 40 = 48$$

$$X = 1.2$$

6.2.2 Calculation with Equations

Calculation of Expected Distance:

$$Distance = SV_{Total} * Offset_{Total} = 1.0 * 100 = 100$$

Solving the Equation:

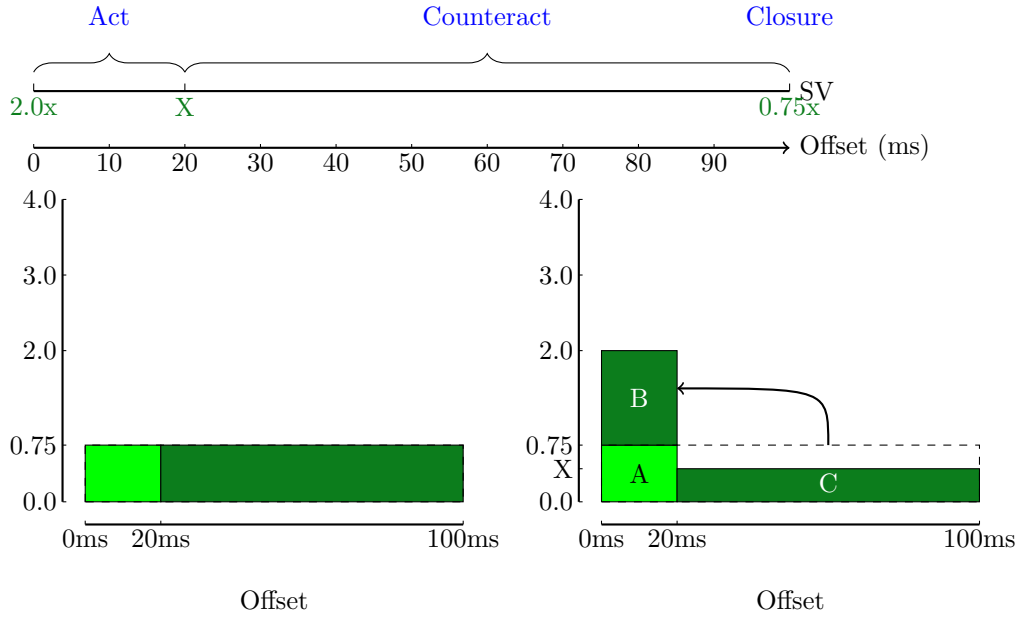
$$SV_1 * Offset_1 + SV_2 * Offset_2 = Distance = 100$$

$$0.2 * 60 + X * 40 = 100$$

$$X = 1.2$$

6.3 Breakdown of an Advanced Effect [3]

Note: We want to have a Visual Spacing with reference to **0.75x**.



No SV Reference *(left)* vs. SV Reference *(right)*

6.3.1 Calculation with Geometry

$$Area_B = SV * Offset = 1.25 * 20 = 25$$

$$Area_B + Area_C = SV * Offset = 0.75 * 80 = 60$$

$$Area_C = (Area_B + Area_C) - Area_B = 60 - 25 = 35$$

$$Area_C = SV * Offset = X * 80 = 35$$

$$X = 0.4375 \approx 0.44(2d.p.)$$

6.3.2 Calculation with Equations

Note the SV_{Total}.

Calculation of Expected Distance:

$$Distance = SV_{Total} * Offset_{Total} = 0.75 * 100 = 75$$

Solving the Equation:

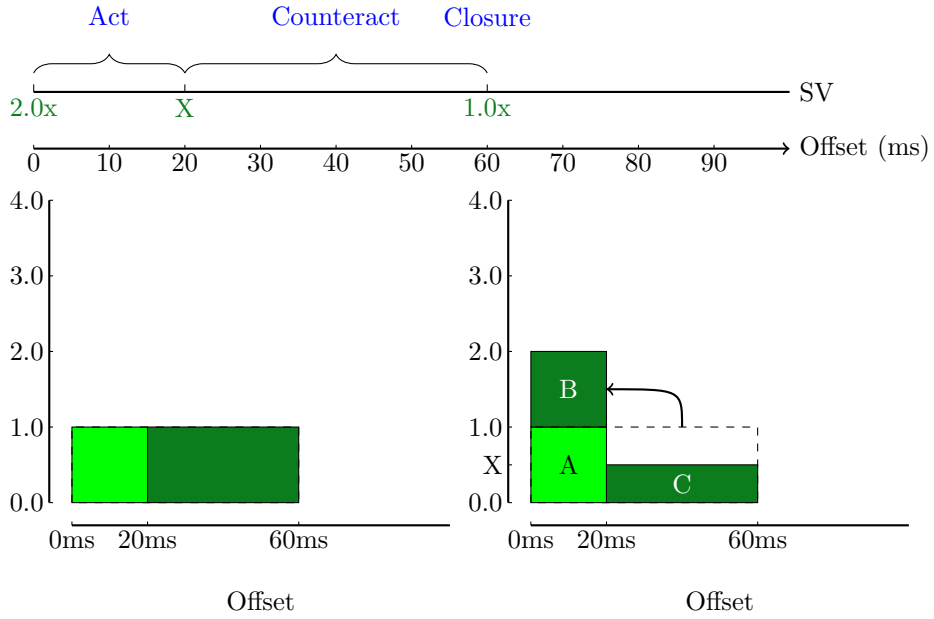
$$SV_1 * Offset_1 + SV_2 * Offset_2 = Distance = 75$$

$$2.0 * 20 + X * 80 = 75$$

$$X = 0.4375 \approx 0.44(2d.p.)$$

6.4 Breakdown of an Advanced Effect [4]

Note: The total offset isn't 100ms.



No SV Reference *(left)* vs. SV Reference *(right)*

6.4.1 Calculation with Geometry

$$Area_B = SV * Offset = 1.0 * 20 = 20$$

$$Area_B + Area_C = SV * Offset = 1 * 40 = 40$$

$$Area_C = (Area_B + Area_C) - Area_B = 40 - 20 = 20$$

$$Area_C = SV * Offset = X * 40 = 20$$

$$X = 0.5$$

6.4.2 Calculation with Equations

Note the $Offset_{Total}$.

Calculation of Expected Distance:

$$Distance = SV_{Total} * Offset_{Total} = 1.0 * 60 = 60$$

Solving the Equation:

$$SV_1 * Offset_1 + SV_2 * Offset_2 = Distance = 60$$

$$2.0 * 20 + X * 40 = 60$$

$$X = 0.5$$

Part VI
Annex

- Agka's SV Tool
- Evening's SV Tool (amber)