Brawl Mechanics: Damage & Knockback Formulae

Cathy J. Fitzpatrick and Chris Immele (Amazing Ampharos)

December 13, 2008

Abstract

In this article, we present an accurate damage formula and an accurate knockback formula. In particular, we develop new results in the areas of how damage is stored, stale move decay, Lucario's aura power, pokemon fatigue, charging smashes, and more. We develop exact models for many of these things that predict the precise damage, including decimal places. We also present a correct formula for knockback.

This article is based on research completed by Cathy J. Fitzpatrick and Chris Immele in December 2008.

1 How Damage is Stored

At least three decimal places of precision are required to explain the exact results we obtained using our damage formula. Whether damage is actually a decimal number is difficult to determine, but whatever it is, it is precise enough to store the equivalent of three decimal places. Contrary to some previous reports, the maximum damage is in fact 999%. Damage over 999% is discarded. The game truncates damage for the purposes of display. For example, 125.651% is displayed as 125%.

2 A Note on Methods

To determine the actual damage a move was dealing, we used it repeatedly and recorded the sequence of numbers after each use. Using a program I wrote, we were able to determine the actual damage the move was dealing, to full precision. At first, we had to work within training mode to avoid stale moves, but once we cracked stale moves precisely, we were able to account for these in the sequence and determine the exact damage of moves directly from sequences of numbers in versus mode.

The source for this program is provided in case other people want to do further research involving decimal damage.

3 The Damage Formula

This formula gives you the exact damage that a move will do in any mode of the game. This formula works for both training mode and versus mode.

$$\begin{aligned} \text{damage} &= \text{base damage} \\ &\times \text{stale multiplier} \\ &\times \text{charge multiplier} \\ &\times \text{aura multiplier} \\ &\times \text{fatigue multiplier} \end{aligned}$$

We explain the value of each of these multipliers in the following sections.

3.1 Base Damage

Every move has some integer damage associated with it. This is its base damage. However, most of the time, the actual damage dealt by a move is not an integer, but this is because of multiplicative factors in the damage formula.

3.2 Stale Move Multiplier

The formula for the stale move multiplier is

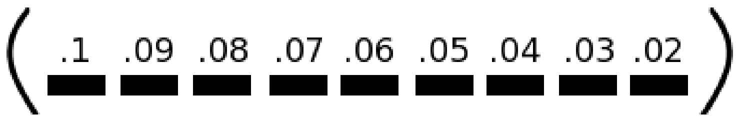
$$\text{stale multiplier} = \begin{cases} 1 & \text{if in training mode or the move is unaffected by stale move decay} \\ 1 - s & \text{if the move is in the stale move queue at least once already} \\ 1.05 & \text{if the move is not in the stale move queue} \end{cases}$$

This is explained below, including the definition of s.

In training mode, stale moves are not in play, so the value of this multiplier is exactly 1. If the move is not affected by stale move decay, then the value of this multiplier is 1 even in versus mode. The following moves are not affected by stale move decay: Luigi's down taunt; DK's cargo throws; tether attacks; and Zero Suit Samus's neutral air.

In versus mode, for moves affected by stale move decay, this factor is never 1. If a move is not in the stale move queue before being used, the value of this multiplier is exactly 1.05.

In versus mode, for moves affected by stale move decay, a move is placed in the stale move queue after it is used. The stale move queue contains nine entries, and each entry has a numerical weighting as depicted in the following diagram.



The left of the diagram represents the most recent move to be added to the queue, and the far right represents the oldest move to the added to the queue. Then we have

$$\text{stale multiplier} = 1 - s$$

where s is the sum of the values of the positions in the queue occupied by the move being used. For example, if the move being used occupies the .1 position, the .06 position, and the .03 position, then the stale multiplier is

$$1 - (0.1 + 0.06 + 0.03) = 0.81$$

3.3 Charge Multiplier

This is the formula for the charge multiplier:

$$\text{charge multiplier} = \begin{cases} 1 & \text{if the move is not a smash attack} \\ 1 & \text{if the smash is uncharged} \\ 1.4 & \text{if the smash is fully charged} \end{cases}$$

For partially charged smash attacks, the value will be somewhere between 1 and 1.4, but we were unable to determine the exact distribution experimentally; it is too difficult to test. However, we speculate that it follows the same distribution as the aura damage multiplier. We speculate that if you charge the smash $^1/_n$ of the way, the multiplier is the same as the aura damage multiplier for $[^{(170-75)}/_n + 75]$ %.

3.4 Aura Multiplier

This is the formula for the aura multiplier:

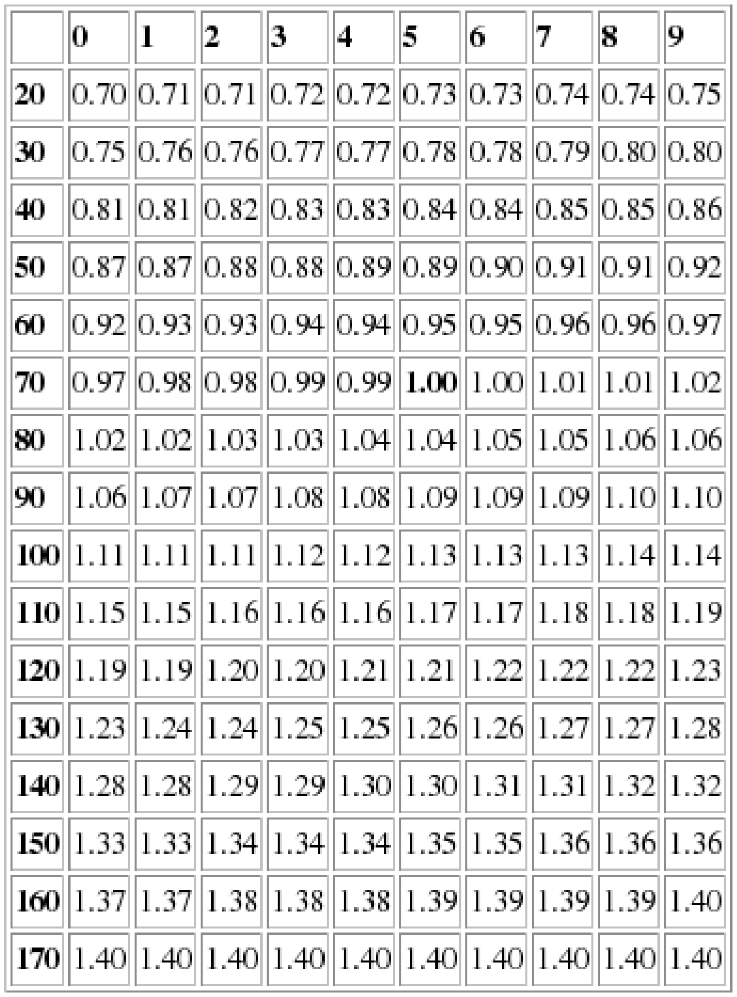
$$\text{aura multiplier} = \begin{cases} 1 & \text{if the move is not an aura move} \\ \text{stock multiplier} \times \text{damage multiplier} & \text{otherwise} \end{cases}$$

where

$$\text{stock multiplier} = \begin{cases} ^6/_7 & \text{if } \Delta\text{stock} \geq +2 \\ ^8/_9 & \text{if } \Delta\text{stock} = +1 \\ 1 & \text{if } \Delta\text{stock} = 0 \\ ^8/_7 & \text{if } \Delta\text{stock} = \text{-}1 \\ ^4/_3 & \text{if } \Delta\text{stock} \leq \text{-}2 \end{cases}$$

In coin mode, each coin counts as a stock for the purpose of the stock multiplier.

The damage multiplier is in the interval [0.7, 1.4]. It is exactly 0.7 for all damages $\leq$ 20% and it is exactly 1.4 for all damages $\geq$ 170%. It is exactly 1 when and only when the damage is 75%. A very good approximation for the damage multiplier can be found by reading the value from this chart:



	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
20	0.70	0.71	0.71	0.72	0.72	0.73	0.73	0.74	0.74	0.75
30	0.75	0.76	0.76	0.77	0.77	0.78	0.78	0.79	0.80	0.80
40	0.81	0.81	0.82	0.83	0.83	0.84	0.84	0.85	0.85	0.86
50	0.87	0.87	0.88	0.88	0.89	0.89	0.90	0.91	0.91	0.92
60	0.92	0.93	0.93	0.94	0.94	0.95	0.95	0.96	0.96	0.97
70	0.97	0.98	0.98	0.99	0.99	**1.00**	1.00	1.01	1.01	1.02
80	1.02	1.02	1.03	1.03	1.04	1.04	1.05	1.05	1.06	1.06
90	1.06	1.07	1.07	1.08	1.08	1.09	1.09	1.09	1.10	1.10
100	1.11	1.11	1.11	1.12	1.12	1.13	1.13	1.13	1.14	1.14
110	1.15	1.15	1.16	1.16	1.16	1.17	1.17	1.18	1.18	1.19
120	1.19	1.19	1.20	1.20	1.21	1.21	1.22	1.22	1.22	1.23
130	1.23	1.24	1.24	1.25	1.25	1.26	1.26	1.27	1.27	1.28
140	1.28	1.28	1.29	1.29	1.30	1.30	1.31	1.31	1.32	1.32
150	1.33	1.33	1.34	1.34	1.34	1.35	1.35	1.36	1.36	1.36
160	1.37	1.37	1.38	1.38	1.38	1.39	1.39	1.39	1.39	1.40
170	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.40	1.40

For example, the damage multiplier for 95% is 1.09.

This chart was created by using quintic interpolation between some exact values that we worked out.

3.5 Fatigue Multiplier

Not everything is known about this multiplier. However, it is 1 if the character is not a pokemon trainer pokemon. Otherwise, it is in the interval [0.7, 1] where 1 is fully unfatigued and 0.7 is fully fatigued. There is in fact a spectrum of fatigue levels and the multiplier can take on various values in between 0.7 and 1. The multiplier cannot take on every value in this interval, however. When the pokemon shows no signs of fatigue, the multiplier is 1.

4 A Note on Training Mode

Our damage formula explains one aspect of training mode that was previously a mystery. In training mode, the stale move multiplier is 1, since stale move decay is not in play. As a result, most moves do their integer base damage always. In versus mode, however, the stale move multiplier is never 1 for moves that respect stale move decay. As a result, moves in versus mode do not usually do integer damage. This is the cause of the inconsistencies noticed between training and versus mode.

5 Knockback

Note: Since the original publication of this article this knockback section has been superseded and supplemented in part by the article "Brawl Dynamics: Velocity, Forces, Knockback" by Cathy J. Fitzpatrick.

The starting place for our research on Knockback was MrSilver's "Character weight list, fall speed list and random things" thread. MrSilver's work is a good starting place, but is incorrect in several ways.

Throughout this section, the words "launch speed" and "knockback" are used interchangeably because the quantity that corresponds to knockback on the results screen is the "max launch(er) speed".

5.1 Constants Associated with Characters

Firstly, for the purpose of calculation of knockback, every character has two associated constants, namely weight1 (w_1) and weight2 (w_2). MrSilver determined both of these constants for every character, but mysteriously referred to weight2 as "fall speed". In fact, weight2 has nothing to do with fall speed. Refer to MrSilver's document for the values of these quantities for each character.

5.2 Constants Associated with Moves

Every move has a base knockback constant (b) and most moves have a knockback growth constant (g). For moves with fixed knockback, the base knockback constant is defined as the knockback taken by Mario. For moves with variable knockback, the knockback growth constant is defined as

$$g = \frac{f - m}{e}$$

where m is the knockback taken by Mario at 0% before the hit, e is any damage you like (say, 100), and f is the knockback taken by Mario at e% before the hit.

Then the base knockback constant is defined as

$$b = m - dg$$

where m is the knockback taken by Mario at 0% before the hit, d is the damage the move deals, and g is the knockback growth constant.

Moves with fixed knockback have have two weight variability constants (s_1, s_2), which describe how much the knockback of the move varies with weight1.

All sufficiently strong moves have a weight2 scale constant (c) which describes how important weight2 is to the move. The constant c can take on a variety of values. For some moves, such as Ness's up throw, c is over 160.

5.3 Knockback Equation

We determined that the knockback equation has two cases depending on the value of a constant k.

To calculate the knockback, first work out k.

$$k = \begin{cases} b + (d + x)\,g/w_1 & \text{if } g \neq 0 \\ s_1 b + s_2 w_1 & \text{if } g = 0 \end{cases}$$

where b is the base knockback constant, d is the damage the move would deal, x is the damage of the target before the hit, g is the knockback growth constant, w_1 is the weight1 of the target, and s_1, s_2 are the weight variability constants.

Then we have the equation for knockback:

$$\text{knockback} = \begin{cases} k + w_2 c & \text{if } k > 2550 \\ k & \text{if } k < 2550 \end{cases}$$

where w_2 is the weight2 of the target, and c is the weight2 scale constant.

5.4 Additional Multipliers for Knockback

We found a few things that cause simple multiplications of the final knockback value. A move being "super effective" is a multiplier of 1.1, and a move being "not very effective" is a multiplier of 0.9. Obviously, this only affects the Pokemon Trainer. In fast mode special brawl, all knockback is multiplied by 1.5, and in slow mode special brawl, all knockback is multiplied by 0.5. Heavy mode special brawl does not affect knockback. As was previously known, attacking a target who is charging a smash other than Ness's up and down smashes is a multiplier of 1.2.

6 Further Research

It is not known how stale move decay affects knockback precisely. Another area for further research is how charging smash attacks affects knockback.