TL; DR

- `MoveData` holds static data about a Move. You can think of it like a Bulbapedia page on the Move.
- `MoveData` should not be confused with `MoveInstance` (which tracks things like cooldown).
- How Move damage and healing are calculated can be seen on page 3.
- Moves have Categories, which can be seen in Table 1.
- Similarly, Moves can make contact (think Spiky Shield), which can be seen in Table 3.

Contents

0.1 What Is `MoveData`?

`MoveData` is an asset that contains static data on a Move. Specifically, it contains information on:

- **Base power**. A measure of the relative power of a Move. Starter Moves may only have 10–30 base power, whereas the most powerful moves may tip the scales at 100. The majority of Moves should have low base powers—we don't want the OHKOs of Pokemon, as this isn't a turn-based game.
- **Base cooldown**. The cooldown in seconds. This gets modified by the `CombatStatsComponent`'s Haste.
- **Category**. A representation of the kind of Move this is. The `MoveCategory`s are:

Table 1: All `MoveCategory`s currently implemented.

Name	Description
`None`	Default. Don't use this. If you see it, it's an error.
`PhysicalDamage`	If damaging, the Move relies on the user's physical abilities, like strength.
`SpecialDamage`	If damaging, the Move relies on the user's non-physical abilities, like psychic or spiritual ability.
`PhysicalHealing`	A move that restores health (self, allies, enemies, etc.) by physical means. For example, an injection, food, a bandage, or acupuncture.

Continued on next page

Table 1: All MoveCategorys currently implemented. (Continued)

SpecialHealing	A move that restores health (self, allies, enemies, etc.) non-physically. For example, mental healing, emotional support, spiritual peace, or the reversing of time to heal wounds.
Summoning	A move that summons an object or creature.
Utility	Does no damage to opponent or self. May still attach effects. Pokemon example: Thunder Wave or Pay Day (but without damage).

- **Contact**. Determines what kind of contact (if any) occurs when the Move hits a target. The MoveCategorys are:

Table 2: All MoveContacts currently implemented.

Name	Description
None	No contact (e.g., Summoning or damaging from afar).
PhysicalContact	The damage or status requires the user to make physical contact with the target. Example: punching something.
SpecialContact	The damage or status requires the user to make non-physical contact with the target. Examples: direct psychic connection, a sustained electrical current from the user to the target, or a "drain life" effect.

  I'll concede that SpecialContact probably isn't super useful to track, but you never know. It's easier to put it in now rather than later.

- **Display name**. If the field DisplayName is blank, the asset's name will be used. May be useful for special characters.

- **Effects to inflict**. A TArray of EffectsToImplements. This includes the Effect, its percentage of being applied on a successful Move hit, and percentages of how many stacks to apply. For example, upon successful hit, a Move may have a 50% chance to inflict 1 stack of a Debuff, a 25% chance to inflict 2 stacks of the Debuff, and a 25% chance to inflict nothing at all.

- **Mutual Effects**. If true, only up to one Effect in EffectsToImplement may be attached. If false, each Effect's probability is checked individually, resulting in (possibly) multiple Effects being attached.

- **Randomness**. A range of values to multiply to calculate the final damage. Normally, any Move's damage is multiplied by a number between 0.85 and 1 (range taken, of course, from Pokemon). This variance makes it a little more exciting.

- **Supporting text**. SupportingText includes dev notes that doesn't appear in-game, a proper description, and flavor text (e.g., for hovers).

- **Move type(s)**. The incredibly vast majority of moves should only associate with one Type. It gets far too complicated and unpredictably unfun to have multi-type attacks.

0.2 Existing MoveData

Table 3: All (non-dummy) MoveDatas currently implemented. (Recall that dummy assets are only used for unit tests and do not show up in-game at all.)

Name	Type	Base Power	Effects	Notes
Blip	Electric	20		

0.3 Damage Calculation Formula

The formulas for damage and healing are inspired by Pokémon. Like it or not, that's the gold standard and what most players will expect when judging base powers, stat values, etc.

The formulas are implemented in CombatStatsComponent.cpp:

$$\text{Damage} = \left[(0.4 \times \text{Level} + 2) \times \frac{\text{Base Power}}{50} \times \frac{\text{Attack}}{\text{Defense}} + 2\right] \times \text{Crit Multiplier} \times \text{Random Fluctuation} \times \text{STAB Multiplier} \times \text{Type Advantage} \times 3^{\lfloor \frac{\text{Level}}{10} \rfloor}$$

where:

- Level is the attacker's Level
- Base Power is determined by the MoveData
- Attack and Defense are the appropriate stat values. Note that these are the calculated stat values, not base values.
- Crit Multiplier is 1.5× if a critical hit occurred; 1.0× otherwise
  - As outlined in the Critical Hit section of the Stats documentation, if a crit does occur, this reverts Defense to its un-buffed value (including everything, e.g., Mutations).
  - If the Critical Hit stat is above 100%, the critical damage increases accordingly. For example, if Crit is 120%, the damage bonus is $1.5 + 0.2 = 1.7\times$.
  - This can be a dangerous game mechanic and will require a lot of balancing. However, it also opens strategies in an otherwise traditional stat scheme.
- Random Fluctuation is determined by the "randomness" of the MoveData (in particular, the field MoveData.RandomRange). Nominally, this is 0.85–1.
- Same Type Attack Bonus (STAB) rewards Monsters for using Moves that share their Types (e.g., a Fire-type Monster using a Fire-type Move). Nominally, this is 1.5×, but can be more or less (e.g., for Typeless Monsters, it's 2×).
- Type Advantage is outlined in the Types documentation. Normally, it ranges anywhere from −1 (Electric actually heals Electric) to 0.25× (double resisted) to 4× (doubly weak). It follows established Pokemon conventions.

Similarly, the healing formula is:

$$\text{Healing} = \left[(0.4 \times \text{Level} + 2) \times \frac{\text{Base Power}}{50} \times \frac{\text{Attack}}{\text{Reference Stat}} + 2 \right] \times \text{Crit Multiplier} \times \text{Random Fluctuation} \times \text{STAB Multiplier} \times 3^{\lfloor \frac{\text{Level}}{10} \rfloor}$$

Here, Reference Value is a dummy `StandardStat` (like Physical Defense, for example) with:

- Base Stat set to 100
- Base Pairs set to 100
- Level set to the defending Monster's Level

0.4 Damage Calculation Case Studies

Here, we analyze an attacking Monster versus a defending Monster. For simplicity:

- Both Monsters had perfect Base Pairs in all stats
- Both Monsters Levels are equal (so a Level 52 attacker goes against a Level 52 defender)
- Crit chance was 0%
- Random fluctuations were 1.0
- STAB and type advantage multipliers were 1.0

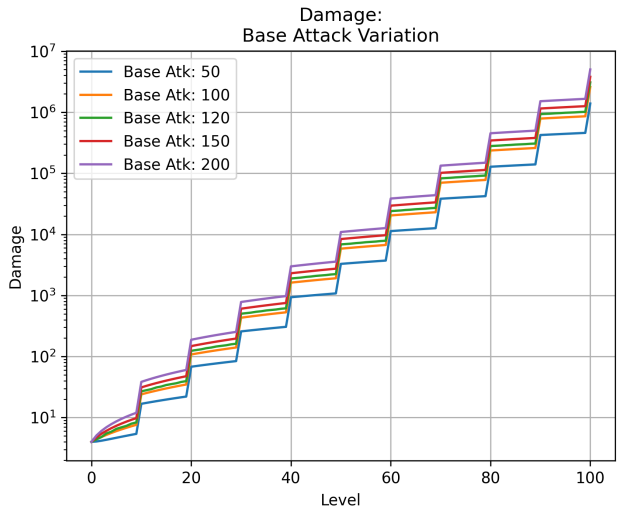

Figure 1: Damage calculation at various Levels and Base Attack stats. Defender had 100 Base Defense and the Move had a base power of 50.

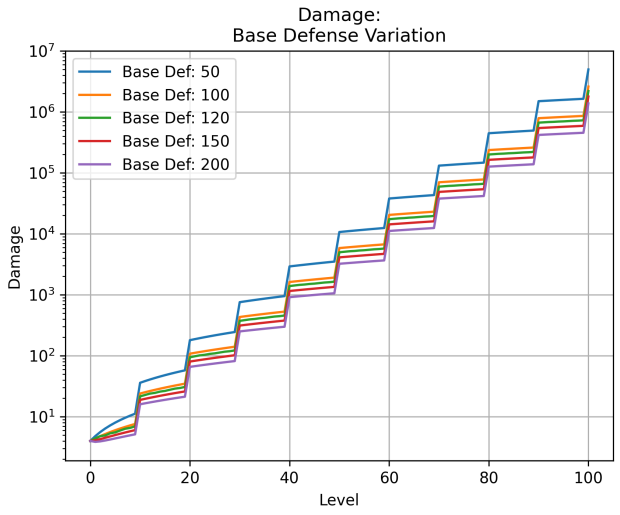

Figure 2: Damage calculation at various Levels and Base Defense stats. Attacker had 100 Base Attack and the Move had a base power of 50. Not surprisingly, this was the inverse of the previous plot.

Okay, so at Level 100, they're dealing, like, 5 million damage. More interestingly, let's vary the base power, leaving the base stats alone:

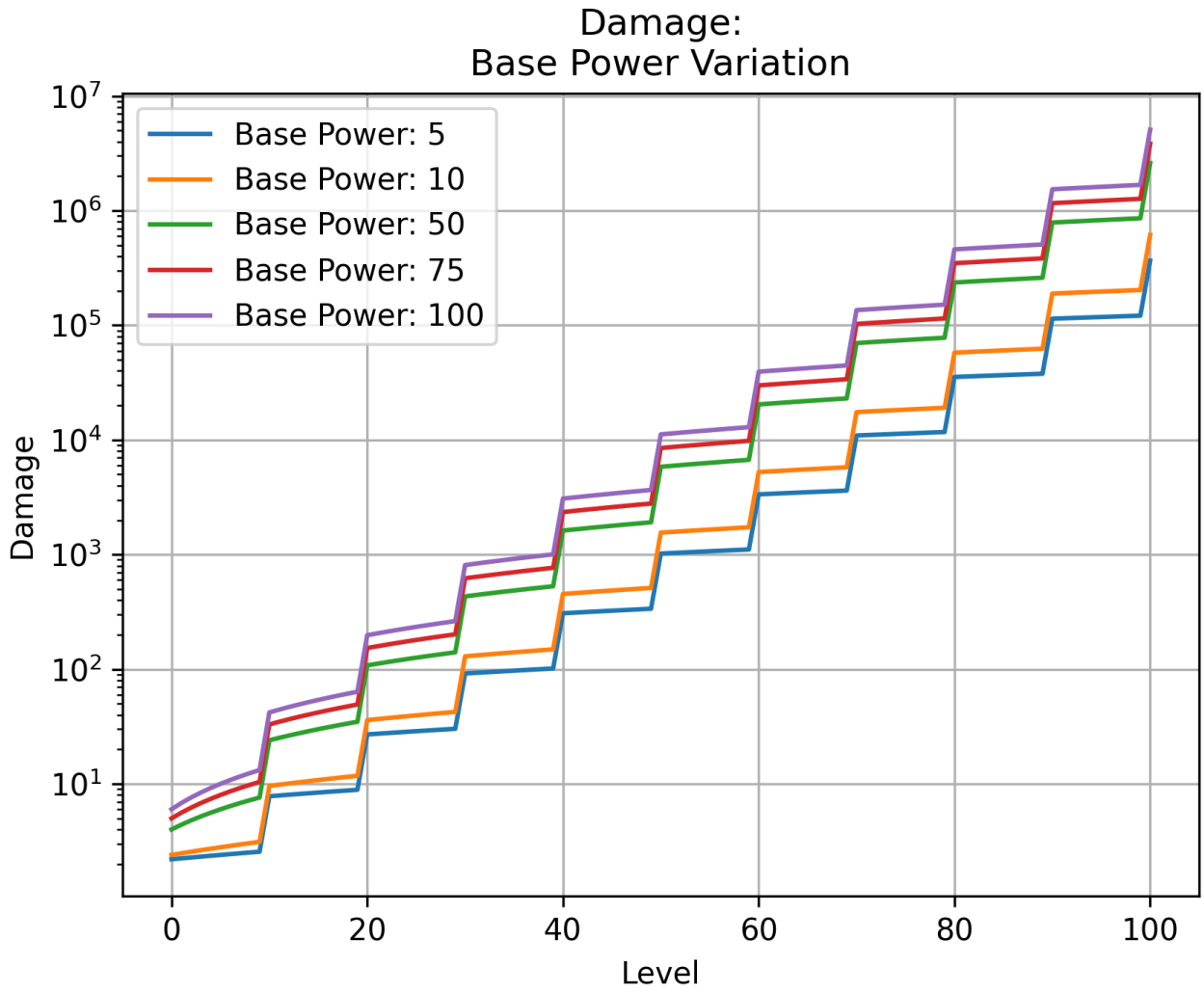

Figure 3: Raw damage calculation at various Levels and base powers. Both attacker and defender are at the same level. Remember: we want a "good" endgame Move to have 50 base power.

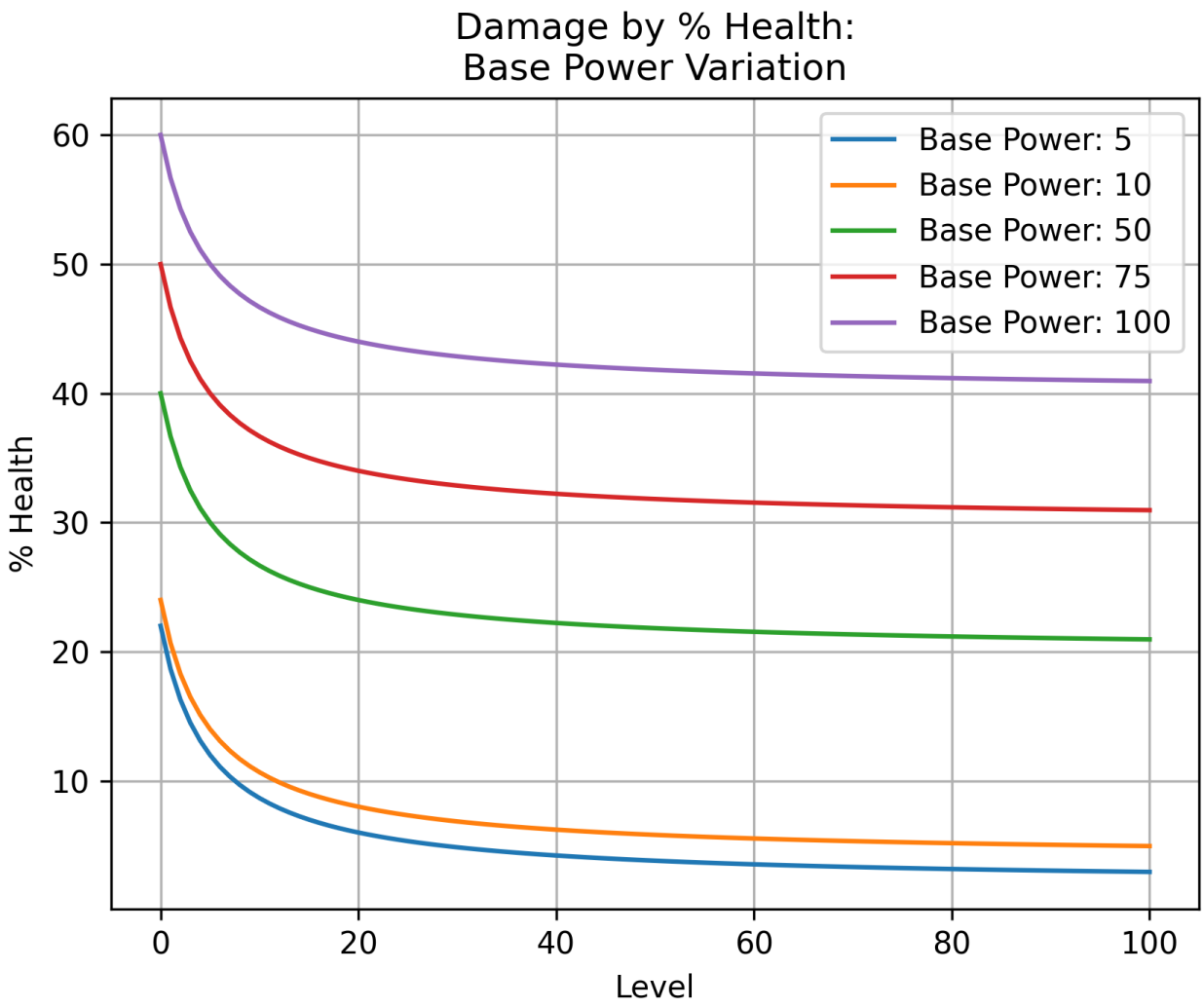

Figure 4: This time, damage is represented as the percentage of the defender's HP. Both attacker and defender are at the same level. Initially, we have fast games, but as the player gets into it, the games slow down and the player has to think harder about choices like Type matchups. Remember: we want a "good" endgame Move to have 50 base power.

Lastly, let's look at an underleveled team in the endgame:

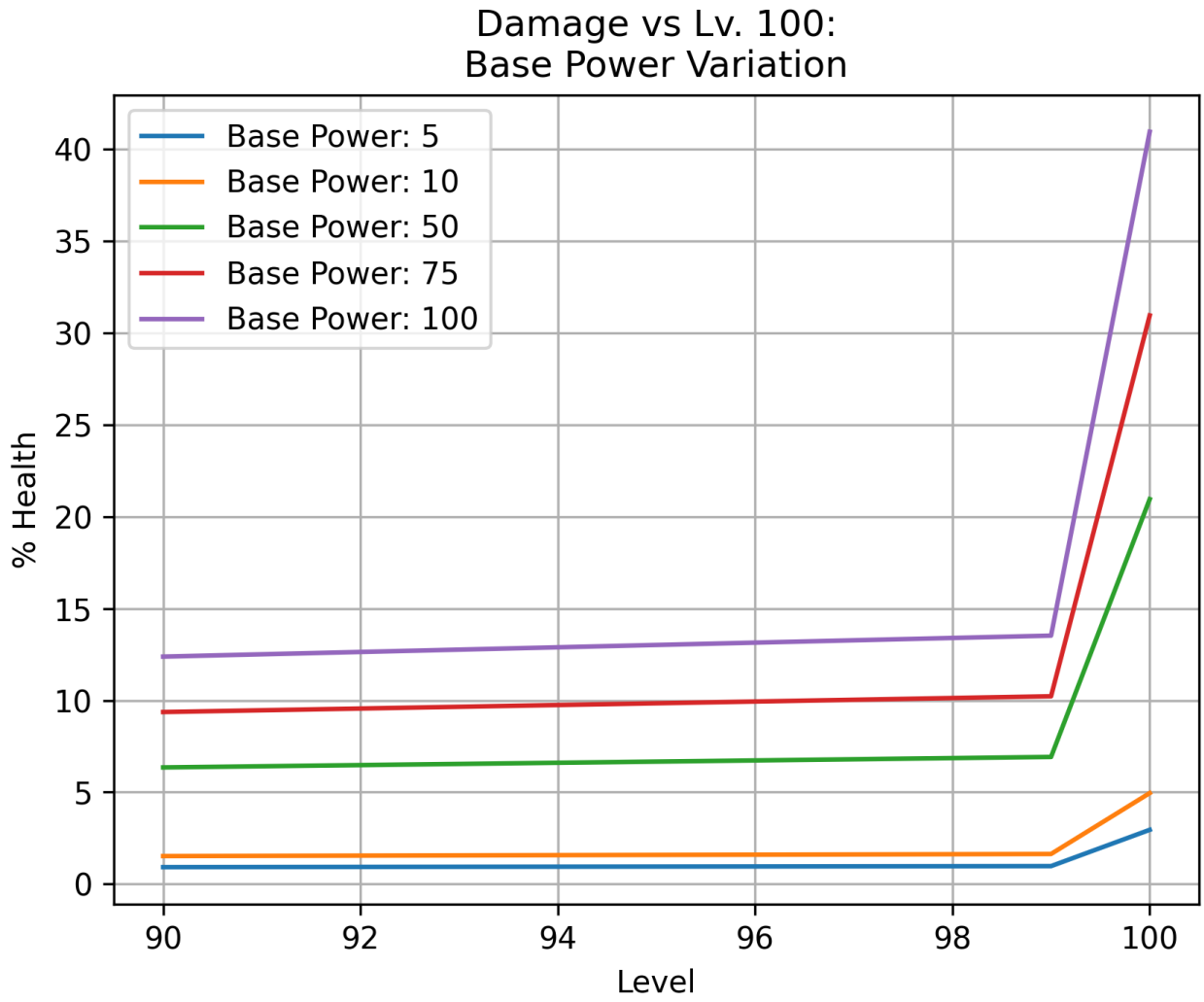

Figure 5: Attacker at various Levels versus a Lv. 100 defender. I think it adds to the sense of accomplishment once a Monster reaches Lv. 100 and you *really* start seeing those big damage numbers. It also makes for a way for grind to make up for lack of skill.