Please note: GPT-4 has been used for graphing within this document only.

osu!taiko pp committee

osu!taiko ppv3

By ltca

C#, MathNET, LINQ
11-23-2024

This document is intended as a general document to explain the aspects of difficulty calculation. It is not intended as precise or detailed.

Contents

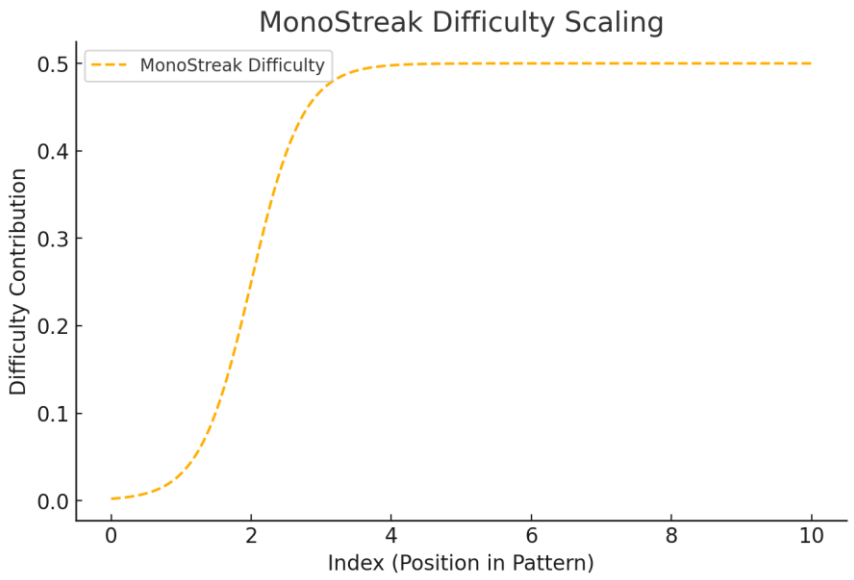
The Colour System in osu!taiko

1. Introduction

In osu!taiko, the colour system evaluates the complexity of rhythmic patterns by analyzing sequences of hit objects. These sequences, consisting of 'don' and 'kat' notes, form patterns that influence gameplay difficulty, particularly in maintaining consistency and reacting to alternating color patterns. The system is structured hierarchically into three primary components: MonoStreaks, AlternatingMonoPatterns, and RepeatingHitPatterns.

2. MonoStreaks: Basic Building Blocks

A MonoStreak represents a sequence of consecutive hit objects of the same type (color). For example, a streak of three 'don' notes forms a MonoStreak, as does a streak of four 'kat' notes.



Attributes of a MonoStreak include:

- HitObjects: The sequence of hit objects within the streak.
- RunLength: The number of hit objects in the streak.
- HitType: The type of the objects in the streak (don or kat).
- Parent: The AlternatingMonoPattern containing this MonoStreak.

The difficulty of a MonoStreak is determined by its position within its parent AlternatingMonoPattern. The difficulty formula is given as:

$$D_{MonoStreak} = \text{Logistic}(e \cdot \text{Index} - 2e) \cdot 0.5$$

3. AlternatingMonoPatterns: Grouping Streaks

An AlternatingMonoPattern groups MonoStreaks of similar run lengths. Alternating patterns represent sequences where players must alternate between colors, such as 'don, kat, don, kat'.

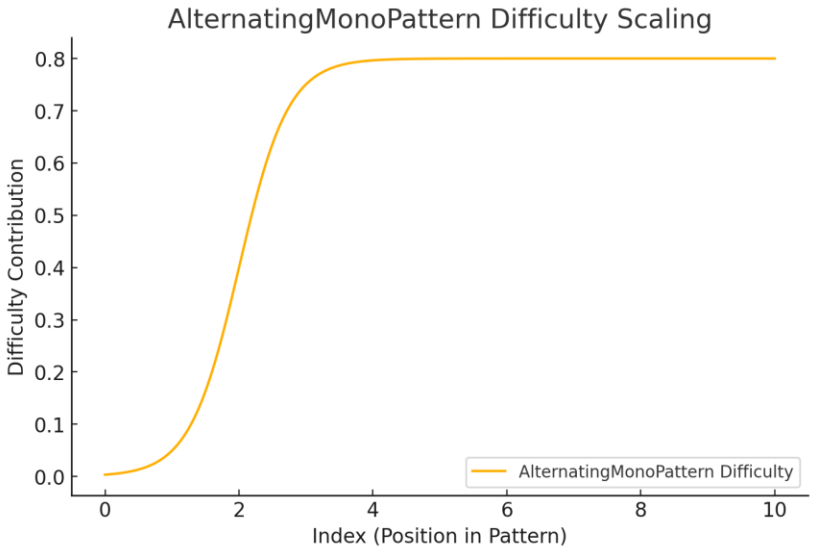
Attributes of an AlternatingMonoPattern include:

- MonoStreaks: A list of MonoStreaks forming the pattern.
- Parent: The RepeatingHitPatterns containing this AlternatingMonoPattern.

- Index: The position of this AlternatingMonoPattern in its parent structure.
- FirstHitObject: The first note in the pattern.

The difficulty of an AlternatingMonoPattern is influenced by its position within the parent RepeatingHitPatterns. The difficulty formula is:

D_AlternatingMonoPattern = Logistic(e * Index - 2e) * D_Parent



4. RepeatingHitPatterns: Identifying Repetition

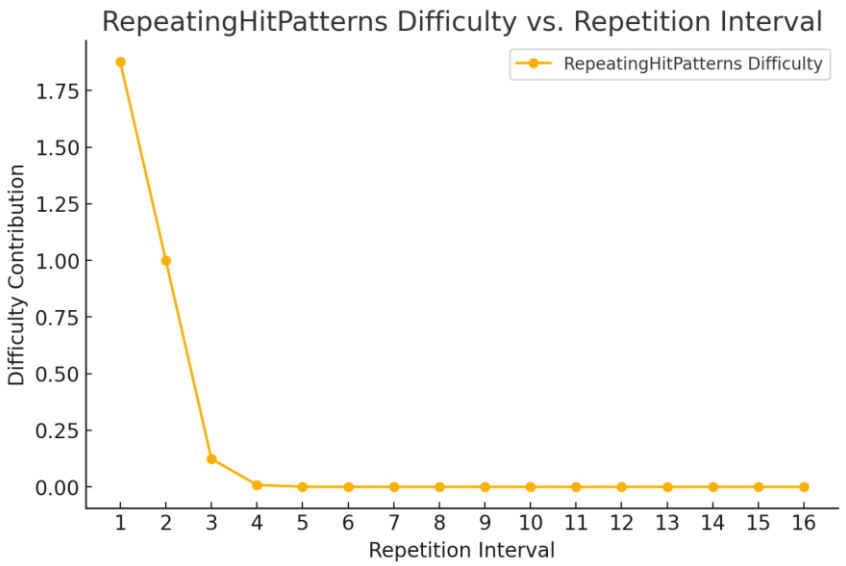
RepeatingHitPatterns capture higher-level repetition within AlternatingMonoPatterns. These structures identify patterns repeated over intervals, increasing difficulty as repetition becomes less predictable.

Attributes of RepeatingHitPatterns include:

- AlternatingMonoPatterns: The patterns grouped into this structure.
- RepetitionInterval: The distance (in patterns) between repetitions.
- Previous: The previous RepeatingHitPatterns in the sequence.

The repetition-based difficulty formula is given by:

D_RepeatingHitPatterns = 2 * (1 - Logistic(e * RepetitionInterval - 2e))



5. Encoding and Preprocessing

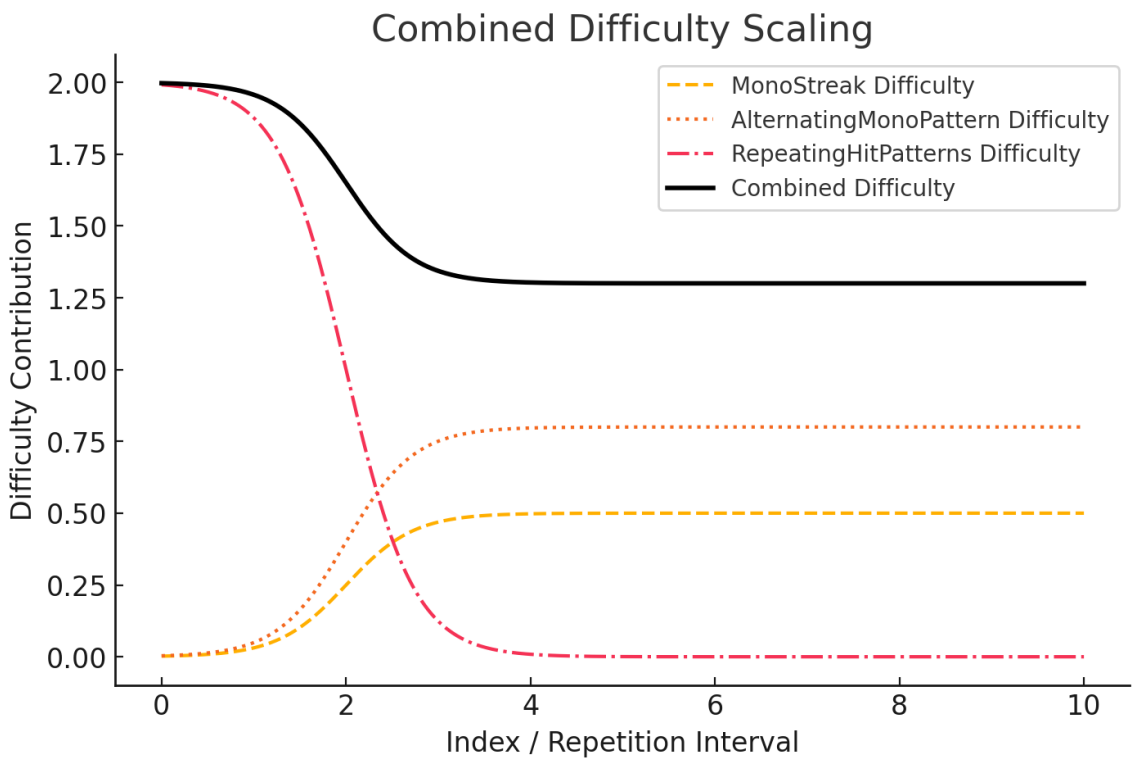
The system encodes hit objects into structured patterns using a multi-step pipeline:

1. MonoStreak Encoding: Groups consecutive notes of the same color.
2. AlternatingMonoPattern Encoding: Groups MonoStreaks based on alternation and length.
3. RepeatingHitPatterns Encoding: Groups AlternatingMonoPatterns based on repetition.

6. Aggregated Difficulty

The ColourEvaluator calculates the total difficulty for a note by aggregating the difficulties of its associated structures:

D_Total = D_MonoStreak + D_AlternatingMonoPattern + D_RepeatingHitPatterns



Marks and Insights

---

8. Theoretical Basis

The osu!taiko colour system is rooted in concepts from rhythm analysis, human perception, and computational complexity:

1. **Pattern Recognition and Memory Load**:
    - The human brain is highly attuned to patterns, and breaking or maintaining a pattern can affect cognitive load. MonoStreaks challenge consistency, while AlternatingMonoPatterns introduce alternating loads, testing a player's adaptability.
2. **Logistic Scaling**:
    - The use of logistic functions ensures smooth, non-linear scaling of difficulty. This reflects diminishing returns at extreme values (e.g., very long

MonoStreaks or distant repetition intervals), which is consistent with how players experience difficulty.

3. **Repetition Detection**:
    - The detection of RepeatingHitPatterns aligns with theories of rhythm salience. Patterns that repeat in predictable intervals create musical motifs, while irregular repetitions demand greater focus.
4. **Hierarchical Structure**:
    - By structuring difficulty into MonoStreaks, AlternatingMonoPatterns, and RepeatingHitPatterns, the system mirrors how players mentally parse complex rhythms into manageable chunks.

The Rhythm System in osu!taiko

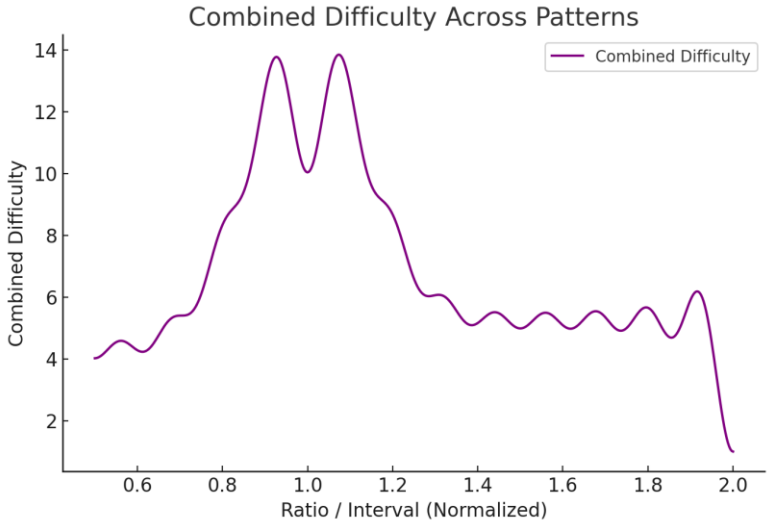
1. Rhythm Structures

1.1 EvenHitObjects

EvenHitObjects group hit objects with minimal or no rhythm variation. For example, a sequence with uniform intervals (e.g., 500ms, 500ms, 500ms) would form a single EvenHitObjects group.

Attributes:

- StartTime: The starting time of the group.
- Duration: The time span from the first to the last hit object in the group.
- HitObjectInterval: The interval between successive hit objects (if consistent).
- HitObjectIntervalRatio: The ratio of intervals between this and the previous group.
- Interval: The time gap between this group and the previous one.

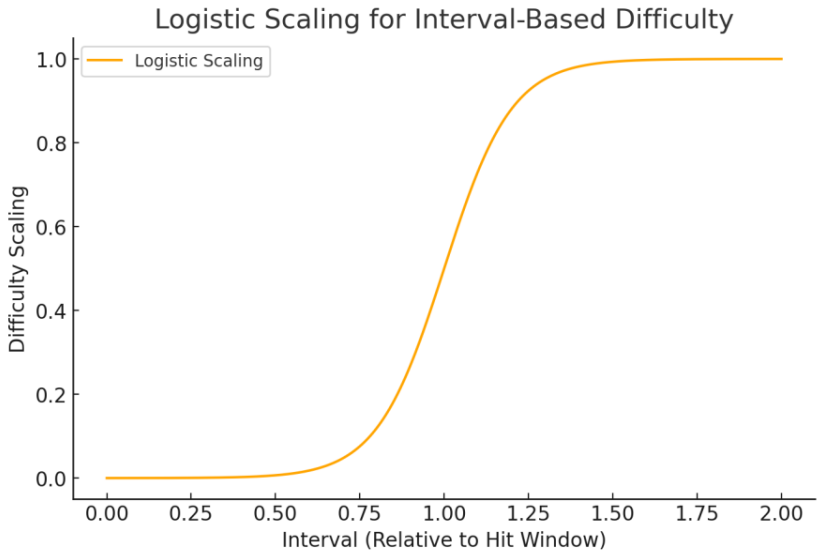


Difficulty Contributions:

1. Interval Changes: Variation in intervals compared to previous groups.
2. Pattern Duration: Penalizing patterns that fit into a single hit window.

1.2 EvenPatterns

EvenPatterns group EvenHitObjects based on similar timing intervals. These patterns represent higher-level structures, capturing repetition and periodicity.

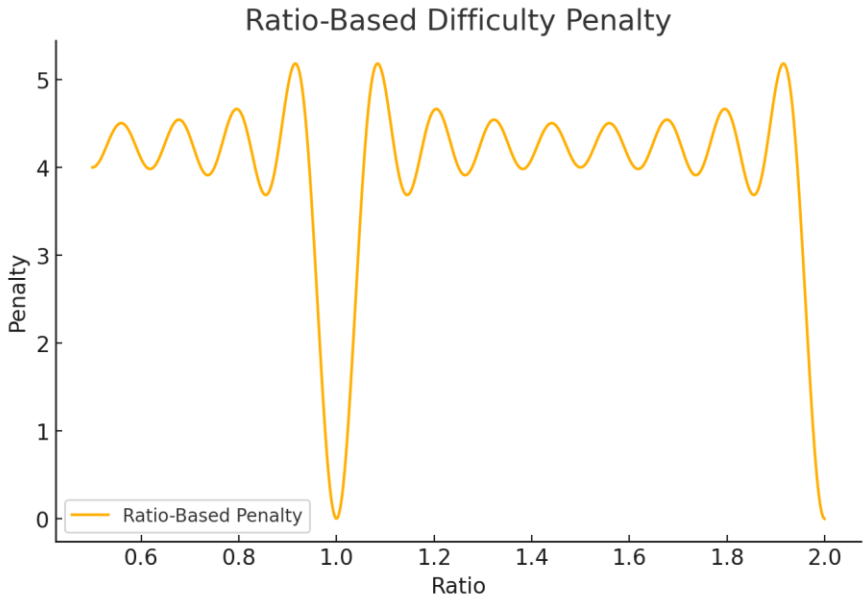


- Attributes:
- ChildrenInterval: The interval between consecutive EvenHitObjects within the pattern.
- IntervalRatio: The ratio of intervals between this pattern and the previous one.
- AllHitObjects: A flattened list of all hit objects in the pattern.

Difficulty Contributions: The difficulty is based on the ratio of intervals between patterns. More significant changes in ratio increase difficulty.

1.3 TaikoDifficultyHitObjectRhythm

This class calculates rhythm-specific difficulty for individual hit objects by comparing their timing with previous hit objects.



- Key Concepts:
- Ratio: The ratio of the current hit object interval to the previous one. Values above 1 indicate a slowdown; values below 1 indicate a speedup.

- Difficulty: A multiplier associated with specific ratio changes. Common rhythmic ratios (e.g., 1:2, 2:1) have predefined difficulties.

Evaluation Methodology: Rhythms are compared to a set of predefined 'common rhythms.' The closest match determines the difficulty.

2. Evaluation and Scoring

The RhythmEvaluator uses ratio-based difficulty to evaluate and score patterns. This includes the following components:

2.1 Ratio-Based Difficulty

3. Penalties for Non-Ideal Ratios:

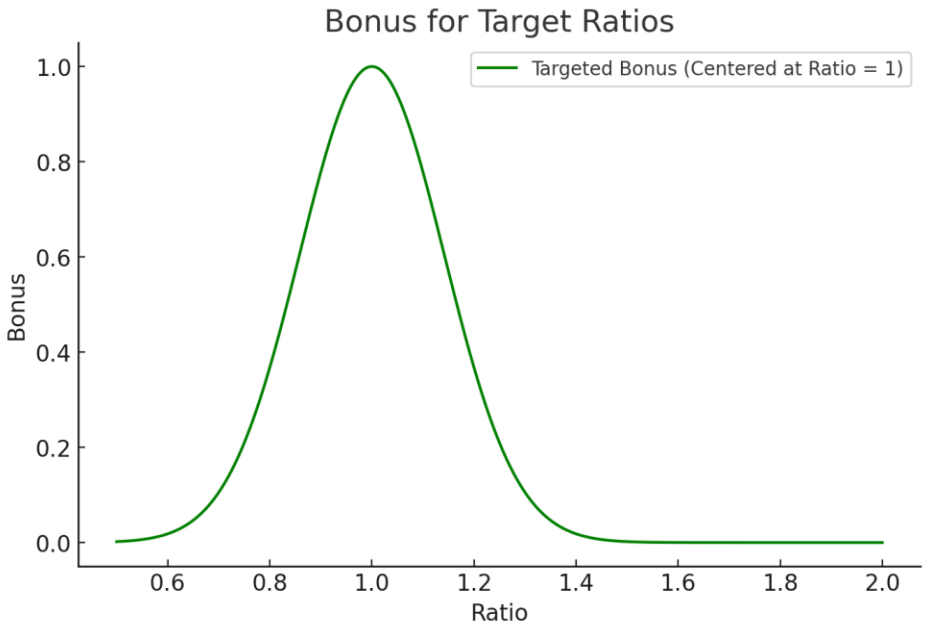
- Penalizes irregular ratios using a periodic cosine function:

*Penalty = - Multiplier * cos(n * π * Ratio)^2*

- Larger denominators incur higher penalties.

4. Bonus for Target Ratios:

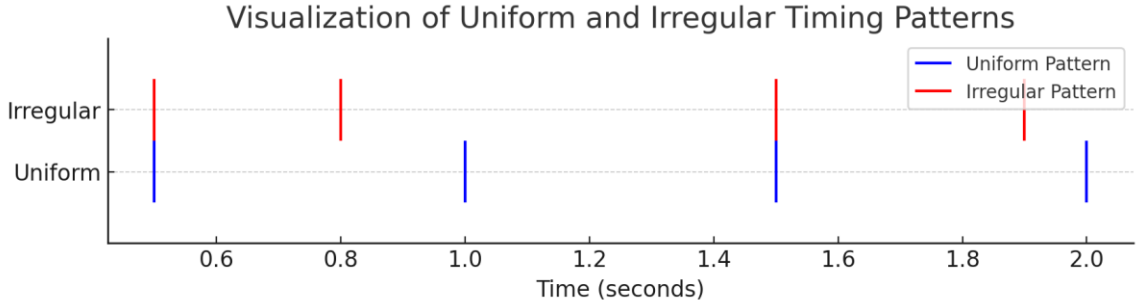
- Provides bonuses for ratios close to 1 (even spacing):



*Bonus = Multiplier * e^(-(Ratio - 1)^2 / Width^2)*

5. Aggregate Difficulty:

- Combines penalties and bonuses for ratios across a pattern.



3. Practical Implications

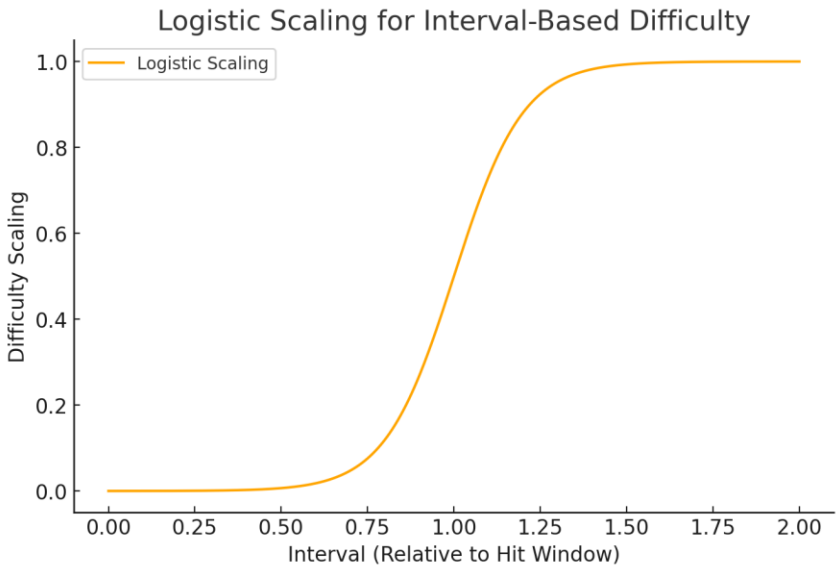
- Gameplay Impact:
- Uniform Patterns: Easier to maintain rhythm.
- Irregular Patterns: Require constant adjustment, increasing cognitive and physical difficulty.
- Speed Changes: Sudden slowdowns or speedups test adaptability.

4. Mathematical Models

Logistic Function for Penalty Adjustment: To ensure difficulty scales realistically, the system applies logistic scaling:

Logistic(x) = 1 / (1 + e^(-k(x - x0)))

Where x0 is the midpoint and k controls the steepness of scaling.



The Stamina System in osu!taiko

1. Core Components

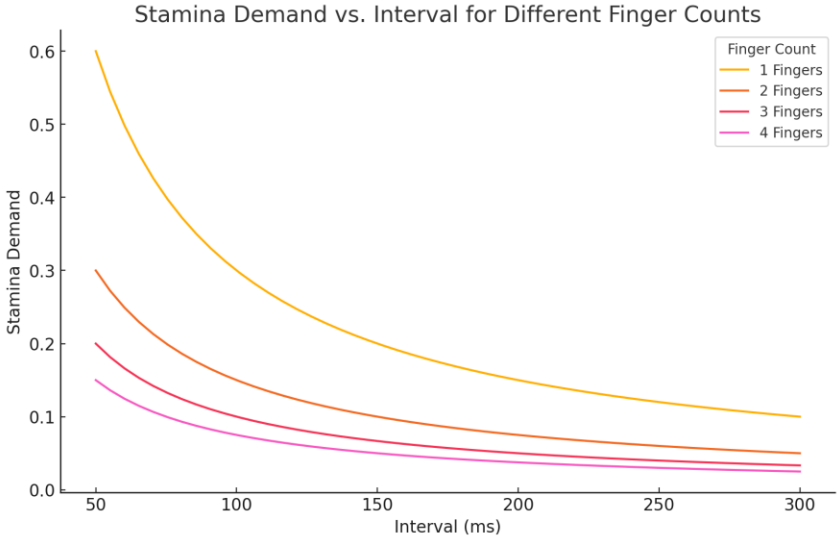
1.1 StaminaEvaluator

The StaminaEvaluator evaluates the difficulty of individual hit objects in terms of stamina demand.

- Key Functions:
- Speed Bonus: Applies a bonus inversely proportional to the interval since the last hit on the same key.

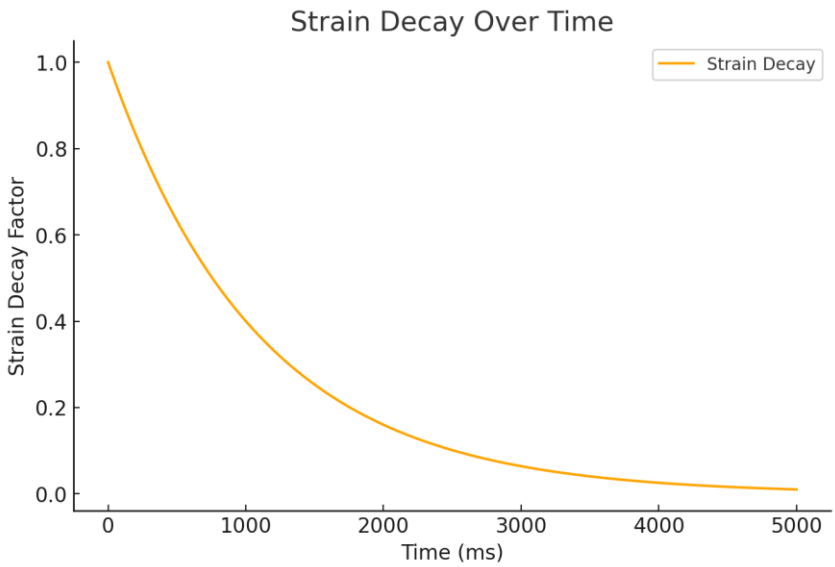
Formula: Speed Bonus = 30 / Interval

- Available Fingers: Determines how many fingers can be used for the current note:



Strain Evaluation: Calculates the strain for a hit object based on:

1. Base Strain: Adds a constant value (e.g., 0.5) to every hit object.
2. Speed Bonus: Adds strain inversely related to the interval since the last hit using the same finger.



1.2 Stamina Class

The Stamina class aggregates strain values over time, incorporating decay and multipliers.

Attributes:

- Skill Multiplier: Scales the strain value (default: 1.1).
- Strain Decay Base: Controls how quickly strain decays over time (default: 0.4).

Core Methods:

- Strain Decay: Applies exponential decay to strain values over time.

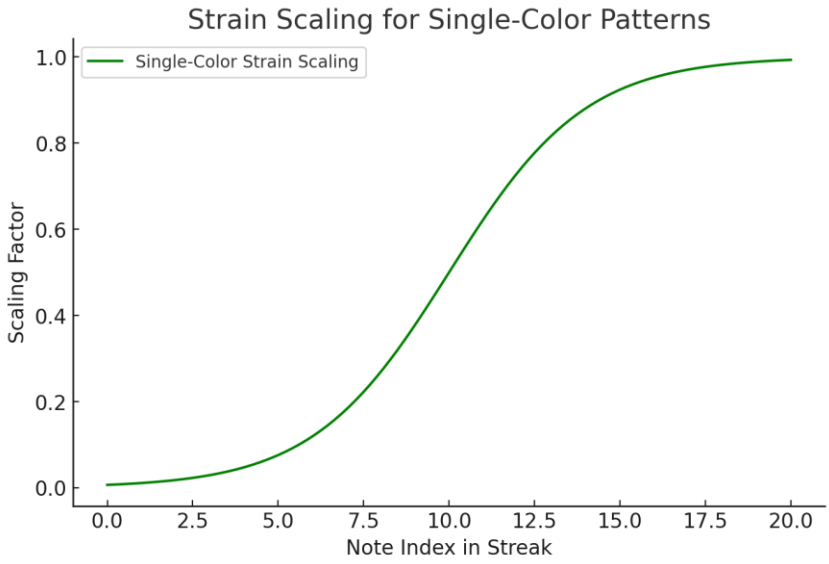
Formula: Strain Decay = Decay Base^(Milliseconds / 1000)

- Strain Value Calculation: Combines decay and new strain contributions.

*Formula: Strain Value = Current Strain * Decay + New Strain * Skill Multiplier*

- Single-Color Patterns: For patterns with consecutive notes of the same color, strain is normalized:

Formula: Strain = Current Strain / (1 + e^(-(Index - 10) / 2))



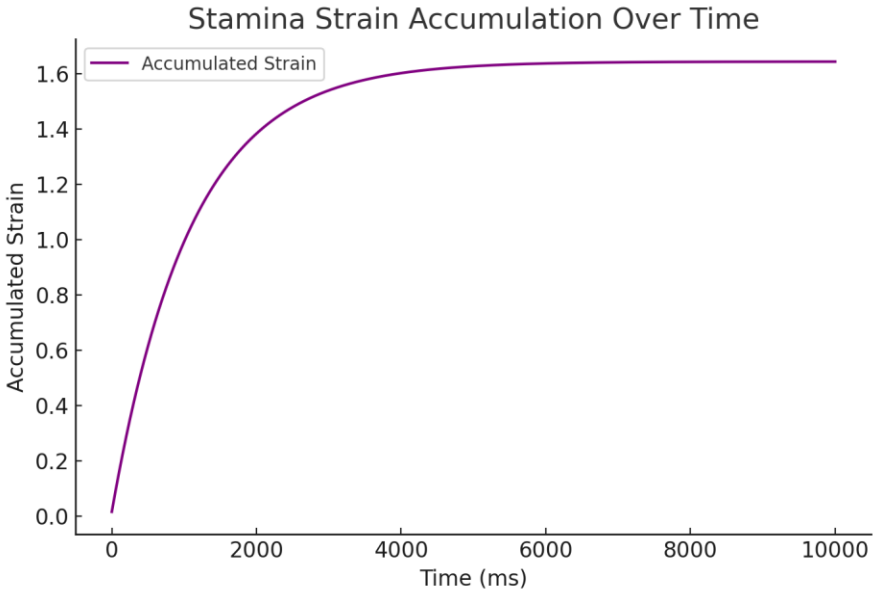
2. Evaluation Methodology

2.1 Stamina Strain Over Time

Stamina strain accumulates during rapid, repetitive hits and decays during breaks or slower sections. This balance ensures that sustained difficulty spikes are reflected in the final difficulty score.

2.2 Color Influence

- The color system interacts with stamina by alternating between don and kat notes:
- Alternating Colors: Allow efficient use of both hands, reducing strain.
- Single-Color Patterns: Force rapid, repetitive use of the same hand, significantly increasing strain.



2.3 Decay Dynamics

Decay prevents older strain contributions from dominating the final difficulty:

*Formula: Strain Decay = Current Strain * Decay Function*

Where the decay function ensures that breaks and slowdowns reduce accumulated strain.

3. Practical Implications

Gameplay Impact:

- Rapid Alternations: Test a player's ability to alternate hands effectively.
- Single-Color Streaks: Stress a single hand, requiring endurance and precise timing.
- Short Intervals: Increase physical strain, particularly in high-speed maps.
- Mapper Considerations:
- Introduce single-color bursts to create stamina challenges.
- Use alternating patterns to balance difficulty and promote rhythmic flow.

Skill Assessment:

- The stamina metric reflects a player's mechanical endurance and speed. It complements rhythm and color metrics by focusing on physical capabilities.

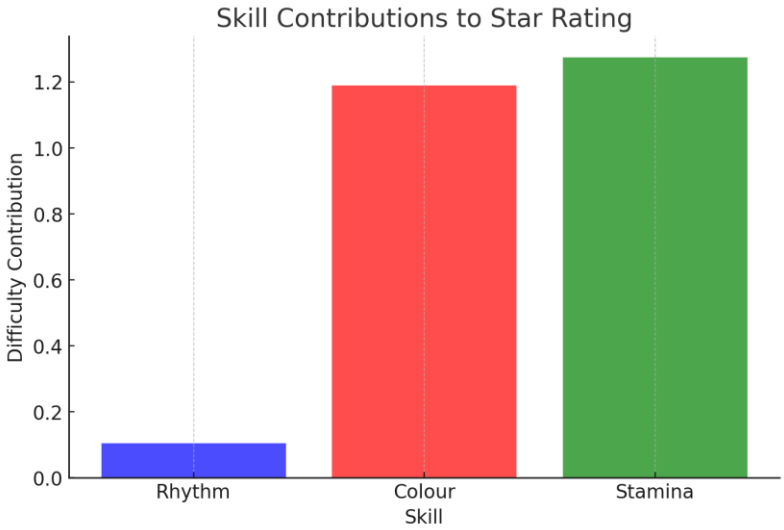
The Difficulty Calculator in osu!taiko

1. Core Components

1.1 Skill Multiplier Weights

The difficulty system uses fixed multipliers for each skill:

- Rhythm Skill Multiplier: 0.07 × Difficulty Multiplier
- Colour Skill Multiplier: 0.425 × Difficulty Multiplier
- Stamina Skill Multiplier: 0.375 × Difficulty Multiplier



The base difficulty multiplier is 0.084375, scaling the overall difficulty.

1.2 Pattern Penalty

A penalty is applied based on the interaction between rhythm and color ratings:

Penalty = log(Threshold / Rating) × min(Upper Bound, log(max(1, Other Rating - Upper Bound)) + Upper Bound)

This penalizes imbalances between rhythm and color contributions. Default threshold: 2.5.

1.3 Strain Decay

The system emphasizes recent difficulty contributions:

Strain Decay = Base Decay^(Milliseconds / 1000)

2. Workflow of the Calculator

2.1 Skill Creation

The system initializes skill components:

- Rhythm: Evaluates timing complexity and irregularity.
- Colour: Analyzes color patterns and alternations.
- Stamina: Calculates physical demand based on tapping speed and repetitions.
- Single-Colour Stamina: Isolates repetitive single-color patterns.

2.2 Difficulty Hit Objects

The calculator preprocesses hit objects to assign color and rhythm attributes:

6. Color Preprocessing: Encodes color groups into MonoStreaks, AlternatingMonoPatterns, and RepeatingHitPatterns.

7. 2Rhythm Preprocessing: Groups hit objects by intervals and regularity using EvenHitObjects and EvenPatterns.

2.3 Skill Ratings

Each skill component outputs a difficulty rating:

Skill Rating = Difficulty Value × Skill Multiplier

Rhythm and color ratings are influenced by penalties for imbalanced patterns. Stamina ratings consider strain accumulation and decay.

Peaks in osu!taiko Difficulty Calculation

1. Importance of Peaks

Peaks represent the most demanding moments in a beatmap, capturing sections with the highest strain values. These peaks are determined by analyzing individual contributions from rhythm, color, and stamina and combining them to reflect overall difficulty.

2. Calculation

Peaks are calculated for each map section by aggregating skill-specific strain values using a p-norm:

Combined Strain = (Σ Strain_i^p)^(1/p)

- n: Number of skills (typically rhythm, color, stamina).
- p: A parameter that emphasizes higher values, typically between 1.5 and 2.

3. Weighted Summation

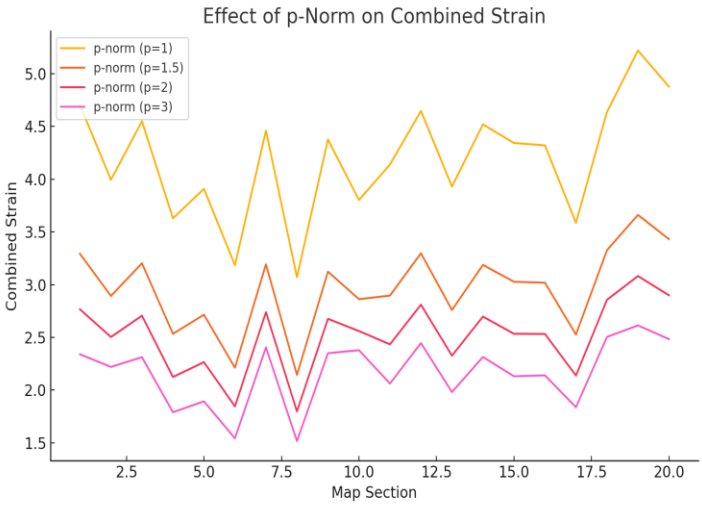
To determine the final difficulty, peaks are sorted in descending order, and a weighted summation is applied:

*Final Difficulty = Σ Peak_i * Weight_i*

- Weight decays exponentially: Weight_i = 0.9^(i-1).

4. Effect of p-Norm on Combined Strain

The graph below demonstrates how different p-norms affect the combined strain calculation across map sections. Higher p-values emphasize peaks, while lower p-values smooth out the values to reflect per-skill strain contributions.



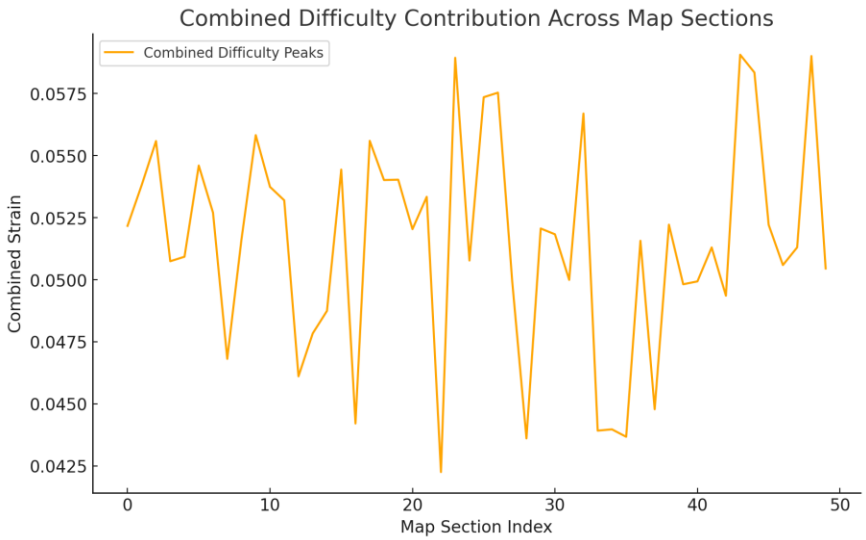
3. Star Rating Calculation

3.1 Combined Difficulty Value

The calculator combines skill ratings using p-norm aggregation:

Combined Strain = ||(Colour Strain, Rhythm Strain, Stamina Strain)||_p

Color and stamina strains are weighted more heavily than rhythm.



3.2 Rescaling

Star ratings are rescaled for consistency:

Star Rating = 10.43 × ln((Raw Rating / 8) + 1)

3.3 Final Adjustments

Specific adjustments are made for edge cases:

- High stamina and low color variance reduce star rating to account for potential abuse of multiple-input playstyles.

4. Practical Implications

Gameplay:

- Combines rhythm, color, and stamina to create balanced maps.
- Encourages mappers to consider how different skills interact across a map.

Skill Assessment:

- Captures both mechanical (stamina) and cognitive (rhythm and color) difficulty.
- Penalizes maps that heavily favor one skill over others.

Reading Skill in osu!taiko

1. Key Components

1.1 Effective BPM and Slider Velocity

The EffectiveBPMPreprocessor calculates the effective BPM and slider velocity for each hit object. This ensures that difficulty calculations reflect variations in scroll speed and map timing.

- Formula:

Effective BPM = Timing BPM × Current Slider Velocity

- Timing BPM: Base BPM from the map's timing points.
- Current Slider Velocity: Adjusted based on scroll speed and clock rate.

1.2 Reading Skill Multiplier

The Reading skill uses a strain decay model to simulate the player's mental load over time.

- Strain Formula:

Current Strain = Previous Strain × Strain Decay Base + Reading Contribution

- Strain Decay Base: Determines how rapidly strain decays (default: 0.4).
- Skill Multiplier: Scales the impact of reading contributions.

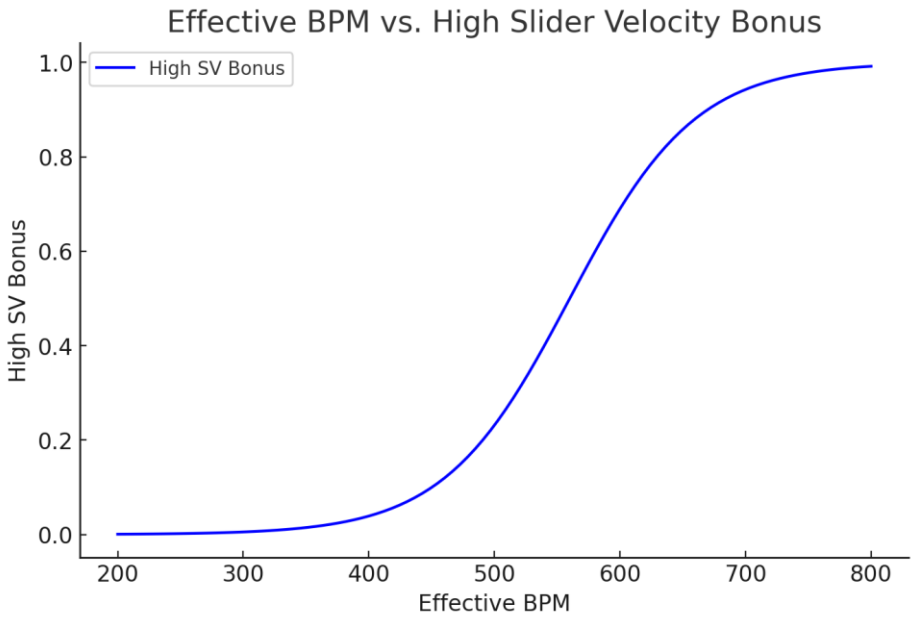
2. Calculations

2.1 High Slider Velocity Bonus

Maps with high slider velocities provide additional strain. This is computed using a sigmoid function:

High SV Bonus = 1 / (1 + e^(-(Effective BPM - Center) / Range))

- Center: Average BPM threshold (default: 560).
- Range: Smooths the transition between low and high bonuses.

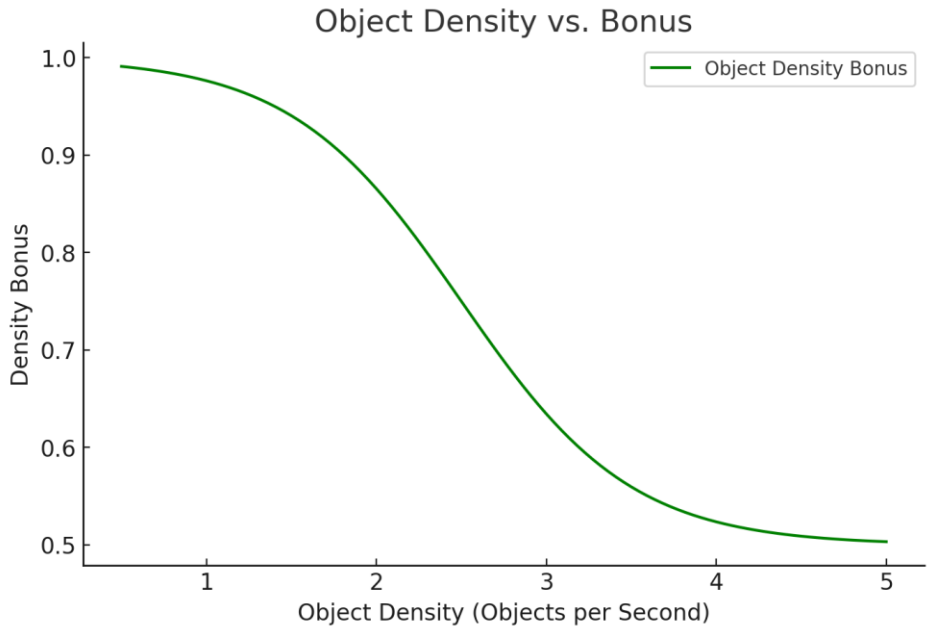


2.2 Object Density

Object density penalizes maps with clustered objects at low scroll speeds. This is also modeled with a sigmoid function:

Object Density = Max Density - (Max Density - Min Density) × Sigmoid Function

- Max Density: Maximum object density bonus.
- Min Density: Minimum penalty for sparse objects.



Final Remarks

With all of these skills combined, morth's system has been effectively removed from the game.

The osu!taiko Performance Calculator

1. Performance Components

1.1 Difficulty Value

The difficulty value reflects the challenge of the map:

*Difficulty Value = ((5 * max(1.0, Star Rating / 0.115) - 4.0) ^ 2.25) / 1150.0*

- Length Bonus scales difficulty with map length:

*Length Bonus = 1 + 0.1 * min(1.0, Total Hits / 1500.0)*

- Miss Penalty reduces difficulty based on effective misses:

*Penalty = Difficulty Value * 0.986^(Effective Miss Count)*

1.2 Accuracy Value

The accuracy value emphasizes precision:

*Accuracy Value = (70 / Estimated Unstable Rate)^1.1 * Star Rating^0.4 * 100*

- Length Scaling adjusts accuracy by map length:

Length Bonus = min(1.15, (Total Hits / 1500.0)^0.3)

1.3 Modifiers

- Modifiers influence both difficulty and accuracy:
    - Hidden: × 1.075
    - Hard Rock: × 1.10
    - Easy: × 0.90
    - Flashlight: Bonus depends on map attributes.

2. Effective Miss Count

The miss penalty accounts for the total hits and scales with shorter maps:

*Effective Miss Count = max(1.0, 1000.0 / Total Successful Hits) * Miss Count*

3. Unstable Rate Estimation

The estimated unstable rate quantifies timing deviation. It is calculated using the proportions of Great and Ok hits, factoring in hit windows. The calculation provides an upper bound for deviation under a 99% confidence interval.

4. Final Performance Calculation

The final performance value combines difficulty and accuracy:

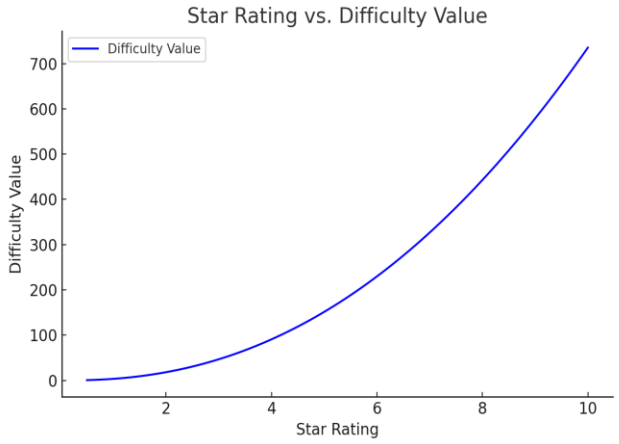
*Performance = ((Difficulty Value^1.1 + Accuracy Value^1.1)^(1/1.1)) * Multiplier*

- Modifiers adjust the final value, including Hidden, Hard Rock, and Easy.

5. Visualizations

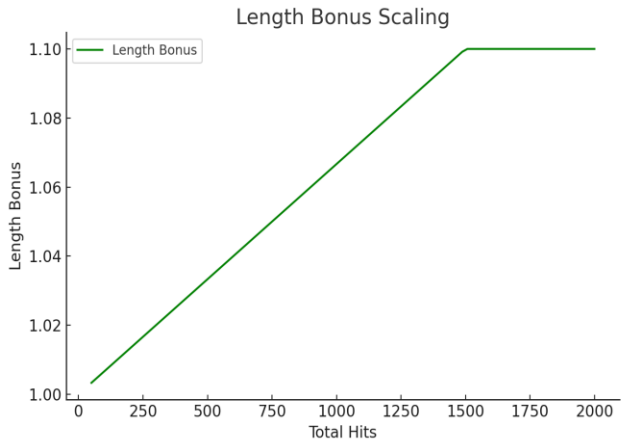
5.1 Star Rating vs. Difficulty Value

This graph shows how difficulty value increases with star rating. The scaling is non-linear, emphasizing the increasing challenge of harder maps.



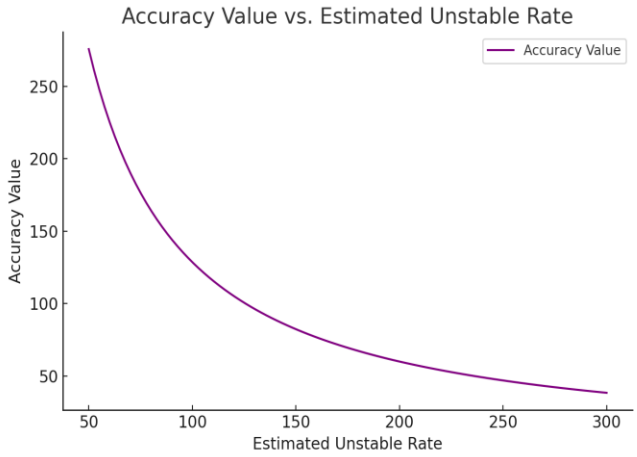
5.2 Length Bonus Scaling

The length bonus increases with the total number of hits, maxing out at 10% for maps with 1500 or more hits.



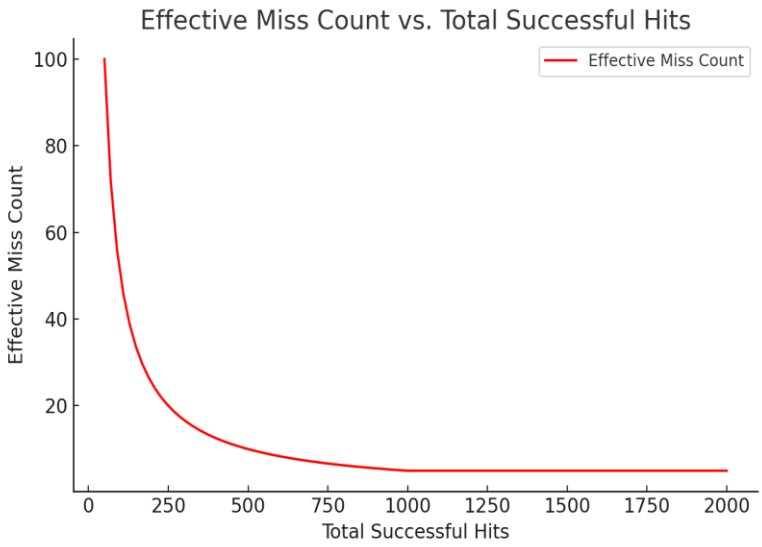
5.3 Accuracy Value vs. Estimated Unstable Rate

Accuracy value decreases as the unstable rate increases, reflecting the importance of precise timing.



5.4 Effective Miss Count vs. Total Successful Hits

Effective miss count decreases as the number of successful hits increases, with penalties scaling more harshly on shorter maps.



5.5 Final Performance Calculation

The final performance value integrates difficulty and accuracy, with star ratings and unstable rates playing a significant role.

Statistical Accuracy in osu!taiko Performance

1. Key Metrics

1.1 Hit Distribution

Accuracy is calculated based on the proportions of Great, Ok, Meh, and Miss hits:

*Accuracy = (300 * H300 + 100 * H100 + 50 * H50) / (300 * (H300 + H100 + H50 + H0))*

- Greats (H300): Perfect hits within the tightest timing window.
- Ok (H100): Hits within a moderate timing window.
- Meh (H50): Hits with a large timing deviation.
- Misses (H0): Notes not hit.

1.2 Unstable Rate (UR)

The unstable rate (UR) represents the standard deviation of hit timing errors. Lower UR values indicate better timing consistency:

*UR = 10 * σ_timing_errors*

2. Timing Deviations

2.1 Deviation Bounds

Deviation bounds are calculated based on hit windows (Great, Ok):

- Great Hit Window (W300): Tight timing window for perfect hits.
- Ok Hit Window (W100): Larger window for acceptable hits.

2.2 Confidence Intervals

To estimate the UR upper bound, the following formula is used:

*UR Upper Bound = W300 / (sqrt(2) * erf^-1(P))*

Where P is the proportion of Great hits, adjusted for confidence.

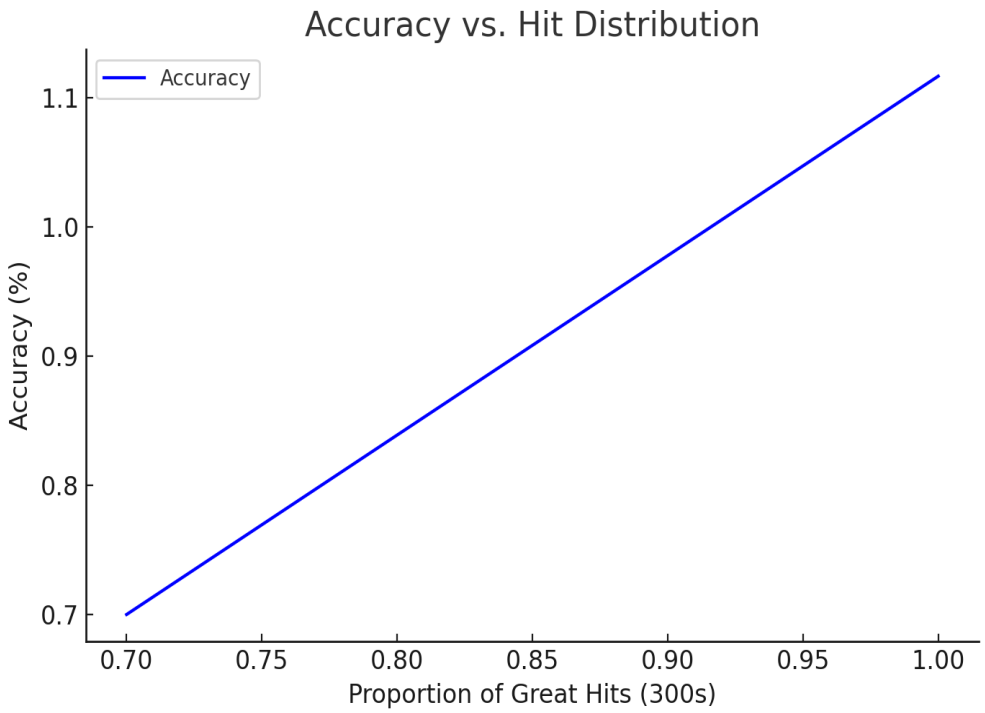
3. Practical Implications

- Gameplay Impact: High accuracy correlates with better performance scores and is critical for competitive play.
- Skill Evaluation: Captures timing precision, complementing rhythm and stamina metrics.
- Performance Scaling: Maps with tighter hit windows demand higher accuracy for competitive PP calculations.

4. Visualizations

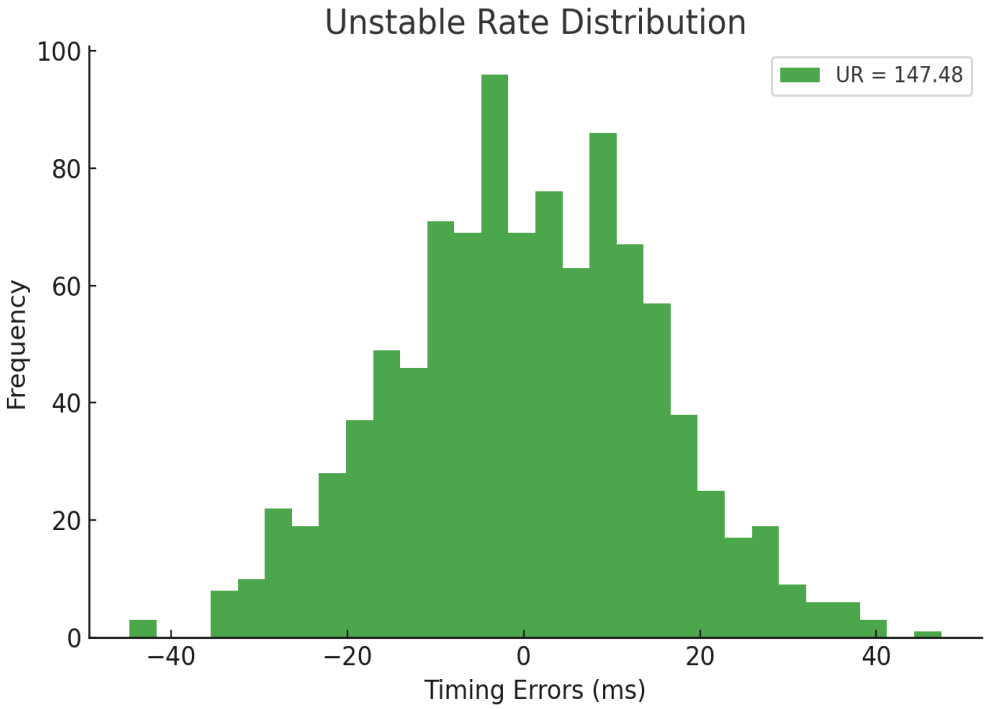
4.1 Accuracy vs. Hit Distribution

This graph shows how the proportion of Great hits (300s) affects overall accuracy. Higher proportions of Great hits yield better accuracy.



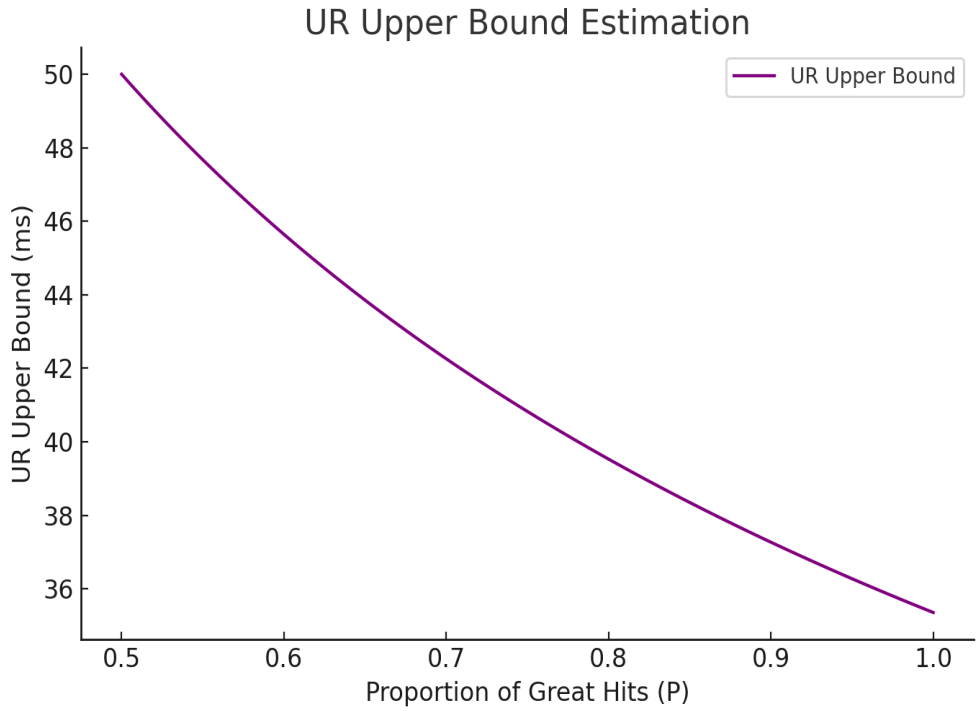
4.2 Unstable Rate Distribution

A histogram of timing errors demonstrates how timing consistency impacts the unstable rate (UR). A narrower distribution corresponds to a lower UR.



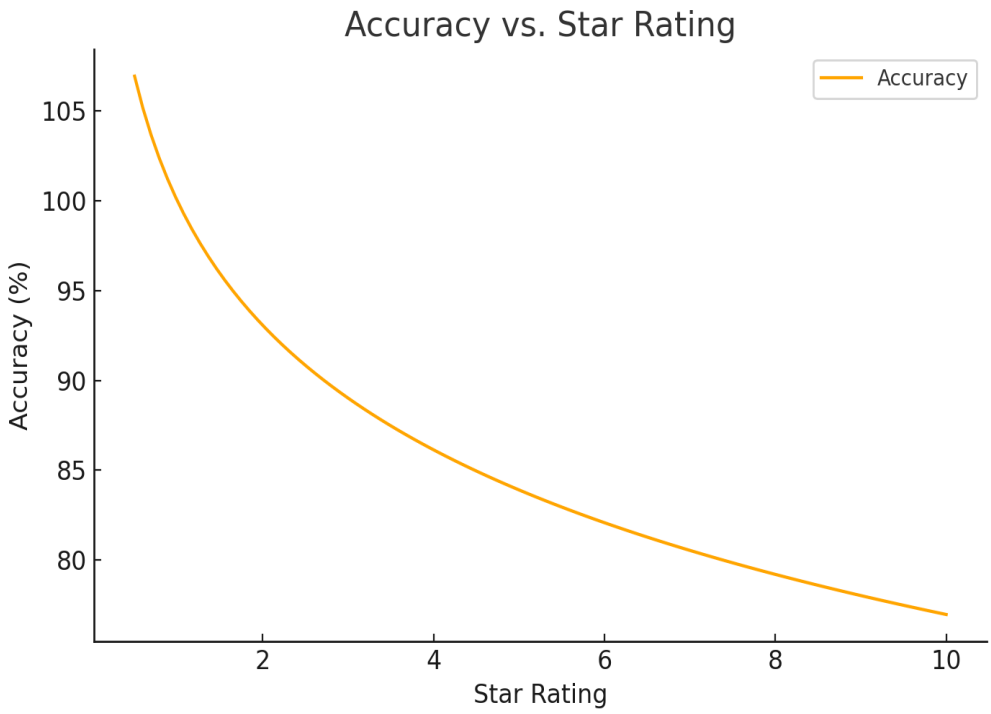
4.3 UR Upper Bound Estimation

The UR upper bound decreases as the proportion of Great hits increases, reflecting greater timing precision.



4.4 Accuracy vs. Star Rating

This graph illustrates the inverse relationship between accuracy and star rating, showing how higher difficulty maps demand more precision.



osu!taiko Attributes Overview

1. Difficulty Attributes

The `TaikoDifficultyAttributes` class defines attributes specific to map difficulty, focusing on skill-based components and timing precision.

1.1 Skill-Based Difficulty

- Stamina Difficulty: Quantifies the stamina required to complete the map.
- Rhythm Difficulty: Reflects the complexity of timing and rhythm patterns.
- Color Difficulty: Measures the challenge of alternating note patterns.

1.2 Peak Difficulty

- Peak Difficulty: Represents the strain of the hardest sections in the map.
- Top Strains:
    - Stamina Top Strains: The most challenging stamina sections.
    - Rhythm Top Strains: The most rhythmically complex sections.
    - Color Top Strains: The hardest sections requiring color alternation.

1.3 Pattern Simplicity

- Mono Stamina Factor: Ratio of mono-color streams to total stamina difficulty.
- Simple Pattern Factor: Penalty applied for maps with overly simplistic patterns.

1.4 Timing Precision

- Great Hit Window: Adjusted timing window for achieving Great hits.
- Ok Hit Window: Adjusted timing window for achieving Ok hits.

2. Performance Attributes

The `TaikoPerformanceAttributes` class focuses on player performance during gameplay, with an emphasis on accuracy and consistency.

2.1 Difficulty and Accuracy

- Difficulty: Reflects the overall difficulty of the map as a combination of its components.
- Accuracy: A percentage-based representation of the player's precision.

2.2 Miss Count

- Effective Miss Count: Penalizes performance based on the adjusted number of misses, incorporating map length and timing consistency.

2.3 Timing Consistency

- Estimated Unstable Rate (UR): Measures the standard deviation of timing deviations, with a lower value indicating better consistency.

osu!taiko Legacy score simulator

1. Scoring Elements

1.1 Swell Ticks

- Description: Each tick within a swell contributes a bonus score.
- Score Contribution: 300 points per tick.
- Combo and Bonus Impact: Does not increase combo but adds to the standardized bonus score.

1.2 Drum Roll Ticks

- Description: Each tick in a drum roll adds bonus points, often affected by mods like Kiai.
- Score Contribution:
    - Base: 300 points per tick.
    - Strong Hits: Add an additional 20% to the base score when applicable.

1.3 Swells

- Description: Large spinners requiring rotations for completion.
- Calculation: The number of required half spins depends on the swell's duration and mods applied (e.g., Double Time or Half Time).

- Each tick in the swell contributes 300 points.
- Bonuses are doubled during Kiai or when the hit object is strong.

1.4 Kiai Time Modifiers

- Description: Active during Kiai sections, providing a 20% increase in bonus score.

2. Formulae for Bonus Calculation

2.1 Bonus Score Ratio

- Determines the ratio of standardized to legacy bonus scores:

Bonus Score Ratio = Standardized Bonus Score / Legacy Bonus Score

- Legacy bonus score accumulates from all bonus hits, while standardized bonus is scaled.

2.2 Scoring Multiplier

- Affects the final score based on applied mods:
  - Easy/No Fail: × 0.5
  - Hidden/Hard Rock: × 1.06
  - Double Time/Nightcore: × 1.12

3. Scoring Example

- Consider a map with:
  - 5 Drum Roll Ticks in Kiai mode (strong hits).
  - 1 Swell requiring 10 rotations.

Bonus Contribution:

- Drum Roll Ticks: 5 × (300 × 1.2) = 1800 points.
- Swell: 10 × 300 = 3000 points.
- Total Bonus: 4800 points (scaled by the mod multiplier).