

Nostalgia 判定机制初步

Yukino

2025 年 10 月 6 日

1 引言

很多游玩过中二节奏的玩家应该都了解判定保护这一概念。这是一种专为中二节奏设计的特殊的判定触发机制，设计初衷是为了避免冗余的按键触发，进而优化游戏的手感。而这种独特的判定机制，也使得玩家以此为基础设计出许多手法，大幅度提升了游戏的可玩性。那么 Nostalgia 作为一款设计类似的无轨音游，自然也具备类似的机制来达到这种判定保护的效果，但实现的方法却有所不同。为了更好的解释这一机制，我们需要对前文一直忽略的 Near 判定进行定义与性质补充，并通过游戏的判定系统分析这些机制是如何实现判定保护的。

2 基础定义

首先定义机制中的基础元素，包括按键、音符、输入及音符状态，所有符号在全文中保持一致。

2.1 按键集合

设 K 表示游戏中所有按键的有限集合。游戏包含 28 个自然序排列的按键，从左到右编号：

$$K = \{1, 2, 3, \dots, 28\}$$

其中每个元素 $k \in K$ 代表一个物理按键，编号反映左右空间顺序（1 为最左按键，28 为最右按键）。

2.2 音符实体

音符是游戏中出现、需玩家输入判定的对象。每个音符定义为元组 $n = (ID_n, U_n, t0_n, T_n)$ ，各分量含义如下：

- $ID_n \in \mathbb{N}^+$ ：音符的唯一标识，用于区分不同音符，避免多音符场景混淆。
- $U_n \subseteq K$ （非空连续子集）：音符对应的“判定成功区域”，即“正确击打”的按键集合。形式化表示为：

$$U_n = \{a_n, a_n + 1, \dots, b_n\}, \quad 1 \leq a_n \leq b_n \leq 28$$

其中 a_n 和 b_n 分别为 U_n 的左边界和右边界。

- $t0_n \in \mathbb{R}_{\geq 0}$ （单位：ms）：音符的“基准时间”，表示音符到达“判定线”的时刻（时间非负）。
- $T_n \subseteq \mathbb{R}_{\geq 0}$ （闭区间）：音符的“有效判定窗口”，定义为基准时间 $t0_n$ 前后各 108ms 的区间。为避免负时间，左边界强制取 0ms：

$$T_n = [\max(t0_n - 108, 0), t0_n + 108]$$

2.3 输入事件

玩家在同一时间内的所有输入对应一个游戏帧，每个输入定义为元组 $I = (ID_I, S_I, t_I)$ ，各分量含义如下：

- $ID_I \in \mathbb{N}^+$ ：输入的唯一标识，区分不同帧的输入。
- $S_I \subseteq K$ 且 $S_I \neq \emptyset$ ：输入中被击打的按键集合。需要注意空集无判定意义。
- $t_I \in \mathbb{R}_{\geq 0}$ （单位：ms）：按键输入发生的时间。同一输入内的所有按键共享同一时间 t_I ，时间精度为一游戏操作帧（16.66ms）。

2.4 音符状态

音符的生命周期由“状态”描述，记为 $State_n$ 。可能的状态集合为：

$$State_n = \{\text{Active}, \text{Unear}, \text{Near}, \text{Miss}\}$$

各状态含义如下：

- 活跃 (Active)：音符未被判定，仍在屏幕上显示。
- 判定成功 (Unear)：音符被正确击打（玩家得分，音符消失）。
- 按歪 (Near)：音符在不正确区域被击打（玩家不得分，音符消失）。
- 超时 (Miss)：音符超时未被判定（玩家不得分，音符消失）。

3 计算

定义音符判定所需的“判定区域”，“距离度量”及“优先级准则”，所有距离计算保留小数精度（至少 1 位小数）。

3.1 判定区域

每个音符包含两个互斥的判定区域：判定成功区域（Unear 区域）和按歪区域（Near 区域）。

3.1.1 判定成功区域（Unear 区域）

音符 n 的判定成功区域 R_n^U 与前文定义的判定成功区域 U_n 完全一致：

$$R_n^U = U_n$$

在 T_n 内击打 R_n^U 中的按键，将触发判定成功（Unear）事件。

3.1.2 按歪区域 (Near 区域)

按歪区域 R_n^N 包含判定成功区域 U_n 左右两侧各 3 个按键范围内的按键，但需排除超出 K 边界及与 R_n^U 重叠的按键。形式化表示为：

$$R_n^N = (\{a_n - 1, a_n - 2, a_n - 3\} \cup \{b_n + 1, b_n + 2, b_n + 3\}) \cap K$$

由定义可知 $R_n^N \cap R_n^U = \emptyset$ (与判定成功区域无重叠)。例如，若 $U_n = \{3, 4, 5\}$ (即 $a_n = 3, b_n = 5$)，则：

$$R_n^N = \{1, 2\} \cup \{6, 7, 8\} = \{1, 2, 6, 7, 8\}$$

3.2 距离与优先级度量

为解决一个输入影响多个音符的冲突，我们在此定义三类优先级准则：时间优先级、距离优先级、位置优先级。

3.2.1 判定成功区域中心

判定成功区域 U_n 的中心 C_n 为其左右边界的几何中点：

$$C_n = \frac{a_n + b_n}{2}$$

例如，若 $U_n = \{3, 4\}$ ，则 $C_n = 3.5$ ；若 $U_n = \{5\}$ ，则 $C_n = 5.0$ 。

3.2.2 按键-音符距离

按键 $k \in K$ 与音符 n 的距离 $d(k, n)$ 定义为按键编号与 C_n 的绝对差值：

$$d(k, n) = |k - C_n|$$

距离越小，说明按键越靠近判定成功区域中心。

3.2.3 优先级层级

对竞争同一输入的多个音符，定义从高到低的全序优先级，具体如下：

1. 时间优先级 (P_{time}): 基准时间 $t0_n$ 更早的音符 (先到达判定线) 优先级更高。对任意两个音符 n_1, n_2 :

$$\text{若 } t0_{n_1} < t0_{n_2}, \text{ 则 } P_{\text{time}}(n_1) > P_{\text{time}}(n_2)$$

若 $t0_{n_1} = t0_{n_2}$, 则进入距离优先级判断。

2. 距离优先级 (P_{dist}): 对基准时间相同的音符, 按键与判定成功区域中心距离更小的音符优先级更高。对任意两个基准时间相同的音符 n_1, n_2 :

$$\text{若 } d(k, n_1) < d(k, n_2), \text{ 则 } P_{\text{dist}}(n_1) > P_{\text{dist}}(n_2)$$

若 $d(k, n_1) = d(k, n_2)$, 则进入位置优先级判断。

3. 位置优先级 (P_{pos}): 对基准时间和距离均相同的音符, 判定成功区域中心更靠左 (C_n 更小) 的音符优先级更高。对任意两个基准时间、距离均相同的音符 n_1, n_2 :

$$\text{若 } C_{n_1} < C_{n_2}, \text{ 则 } P_{\text{pos}}(n_1) > P_{\text{pos}}(n_2)$$

4 判定事件

判定事件是“输入-音符”映射的结果, 需要注意到每个事件间互斥 (一个音符仅能触发一个事件)。

4.1 判定成功事件 ($E_n^U(I, k)$)

当按键击打音符的判定成功区域时, 触发判定成功事件。

触发条件 (需同时满足):

1. 状态有效: $State_n = \text{活跃 (Active)}$ (音符未被判定)。
2. 时间有效: $t_I \in T_n$ (输入时间在音符判定窗口内)。
3. 区域有效: $k \in S_I \cap R_n^U$ (按键 k 被击打且属于判定成功区域)。
4. 按键有效: k 未被其他音符的判定失效 (见第 6 节)。

事件效果:

- 更新音符状态: $State_n = \text{判定成功 (Unear)}$ (音符消失, 玩家得分)。
- 当前输入帧内, 按键 $k \in S_I \cap (R_n^N \cup R_n^U)$ 失效, 当且仅当集合中的所有按键无法触发判定成功事件。

4.2 按歪事件 ($E_n^N(I, k)$)

当按键击打音符的按歪区域 (未击中判定成功区域) 时, 触发按歪事件。

触发条件 (需同时满足):

1. 状态有效: $State_n = \text{活跃 (Active)}$ 。
2. 时间有效: $t_I \in T_n$ 。
3. 区域排除: $k \notin \bigcup_{m \in \text{音符集合}} (S_I \cap R_m^U)$ (按键 k 未击中任何音符的判定成功区域)。
4. 区域有效: $k \in S_I \cap R_n^N$ (按键 k 属于按歪区域)。
5. 按键有效: k 未被其他音符的判定失效。

事件效果:

- 更新音符状态: $State_n = \text{按歪 (Near)}$ (音符消失, 玩家不得分)。
- 当前输入帧内, 按键 $k \in S_I \cap (R_n^N \cup R_n^U)$ 失效, 当且仅当集合中的所有按键无法触发判定成功事件。

4.3 超时事件 (E_n^M)

音符超时未被判定成功或按歪时, 触发超时事件。

触发条件 (需同时满足):

1. 状态有效: $State_n = \text{活跃 (Active)}$ 。
2. 时间超时: $t_{\text{当前}} > t_{0_n} + 108$ (当前时间超出判定窗口右边界)。

事件效果:

- 更新音符状态: $State_n = \text{超时 (Miss)}$ (音符消失, 玩家不得分)。

4.4 无判定事件 ($E_n^{\text{None}}(I)$)

输入对音符无任何判定效果时, 触发无判定事件。

触发条件 (满足任意一条即可):

1. 状态无效: $State_n \neq \text{活跃 (Active)}$ (音符已被判定)。
2. 时间无效: $t_I \notin T_n$ (输入时间在音符判定窗口外)。
3. 区域无效: $S_I \cap (R_n^U \cup R_n^N) = \emptyset$ (输入按键未击中音符的任何判定区域)。
4. 按键无效: S_I 中所有按键均被其他音符的判定失效。

事件效果:

- 音符状态保持 活跃 (Active) (继续等待后续输入)。

5 按键冲突时的优先级规则

当一个输入 (或按键) 影响多个音符时, 需通过优先级规则确保唯一判定结果。

5.1 优先级判断流程

对任意输入 I 和活跃音符集合 $\mathcal{N}_{\text{Active}}$, 执行以下步骤:

1. 按“时间优先级 > 距离优先级 > 位置优先级”对 $\mathcal{N}_{\text{Active}}$ 排序, 得到有序列表 $[n_1, n_2, \dots, n_m]$, 其中 n_1 优先级最高。
2. 按列表顺序处理每个音符 n_i :

- (a) 检查输入 I 中未失效的按键，是否存在能触发 $E_n^U(I, k)$ (判定成功事件) 的按键 k 。若存在，触发事件并使 k 失效。
 - (b) 若无按键触发判定成功事件，检查是否存在能触发 $E_n^N(I, k)$ (按歪事件) 的未失效按键 k 。若存在，触发事件并使 k 失效。
3. 当输入 I 中所有按键均失效，或所有活跃音符均处理完毕时，停止流程。

5.2 多键输入的处理准则

同一输入内的按键无固有顺序，优先级规则确保以下两点：

- 用于判定高优先级音符的失效按键，不再影响低优先级音符。
- “无效按键”（未击中任何音符判定区域的按键）被直接忽略，不干扰有效按键的判定。

6 按键与输入失效

输入失效机制确保对于单一输入帧内的一个按键输入，仅能判定一个音符。

6.1 失效按键集合

对输入 I ，定义“失效按键集合” $\text{Inv}(I)$ 为触发过判定成功或按歪事件的按键集合：

$$\text{Inv}(I) = \{k \in S_I \mid k \text{ 触发某音符的 } E_n^U(I, k) \text{ 或 } E_n^N(I, k)\}$$

6.2 失效范围约束

失效机制的范围约束如下：

- 失效具有“帧特异性”： $\text{Inv}(I)$ 中的按键仅在当前输入 I 内失效。在后续输入帧中，相同按键恢复有效性。

- 失效具有“音符无关性”：失效按键在当前输入内，无法用于判定任何其他音符（无论音符优先级高低）。

7 判定的核心原则总结

为确保判定逻辑一致性，机制遵循以下三条核心原则：

1. 判定唯一性原则：一个音符仅能触发一个判定事件（判定成功/按歪/超时），触发后立即消失。
2. 帧独立性原则：不同输入帧相互独立，前一帧的按键失效不影响后一帧的按键有效性。
3. 优先级唯一性原则：所有冲突都可以通过“时间 > 距离 > 位置”的固定层级解决，因此不存在歧义性结果。

8 一种特殊情况：PJust 区间内的重叠音符

当满足特定时间与区域条件时，两个相近的音符可以突破常规失效规则，允许一次输入同时触发两者的判定成功事件。本规则仅在满足以下全部条件时生效，不影响原有判定逻辑。

8.1 前提条件

设存在两个音符 n_a （前序音符）和 n_b （后序音符），需同时满足以下两点：

- 时间窗口约束：两音符基准时间差不超过 PJust 区间（即 $41 \times 2 \text{ ms}$ ）：

$$|t_{0b} - t_{0a}| \leq 82$$

- 区域重叠约束：后序音符的判定成功区域与前序音符的按歪区域存在非空交集：

$$R_b^U \cap R_a^N \neq \emptyset$$

8.2 触发条件

存在输入 $I = (ID_I, S_I, t_I)$ 满足以下两点:

- 输入时间同时位于两音符的有效判定窗口内:

$$t_I \in T_a \cap T_b$$

- 输入按键集合恰好覆盖两音符的判定成功区域, 且包含重叠区域的按键:

$$S_I = R_a^U \cup (R_b^U \cap R_a^N)$$

示例: 若 $R_a^U = \{2, 3\}$, $R_a^N = \{4\}$, $R_b^U = \{3, 4\}$, 则需满足 $S_I = \{3, 4\}$ (其中 $3 \in R_a^U$, $4 \in R_b^U \cap R_a^N$)。

8.3 判定结果

满足上述条件时, 判定结果如下:

- 两个音符同时触发判定成功事件:

$$E_{n_a}^U(I, k_a) \wedge E_{n_b}^U(I, k_b)$$

其中 $k_a \in R_a^U \cap S_I$, $k_b \in (R_b^U \cap R_a^N) \cap S_I$ 。

- 在这时, 失效规则不介入计算。即本次输入中, 参与判定的按键不进入失效集合:

$$\{k_a, k_b\} \cap \text{Inv}(I) = \emptyset$$

即打破前文按键输入消除的常规约束, 允许两音符共享输入按键。

8.4 边界约束

本特殊规则的边界限制如下:

1. 若不满足前提条件中的任意一条, 则自动适用原有判定规则 (按键 k_b 会因 k_a 触发而失效)。

2. 本规则仅适用于两个音符的协同判定，不扩展至三个及以上音符的场景：游戏中基本不存在满足上述规则的三音符谱面（特殊情况：女神的微睡 Real），且四音符的场景与本规则存在时间窗口冲突，故不满足条件（感兴趣的读者可自行证明）。
3. 当协同判定条件与原有优先级规则冲突时，本规则优先级更高。

8.5 示例说明

设输入与音符参数如下：

- n_a : $R_a^U = \{2, 3\}$, $R_a^N = \{4\}$, $t_{0a} = 1000 \text{ ms}$
- n_b : $R_b^U = \{3, 4\}$, $t_{0b} = 1050 \text{ ms}$ （满足 $|1050 - 1000| = 50 \leq 82$ ）
- 输入 I : $S_I = \{3, 4\}$, $t_I = 1030 \text{ ms}$ （位于 $T_a \cap T_b$ 内）

判定结果如下：

- n_a 通过按键 $3 \in R_a^U$ 触发 $E_{n_a}^U$
- n_b 通过按键 $4 \in R_b^U \cap R_a^N$ 触发 $E_{n_b}^U$
- 按键 3, 4 均不失效，同时完成两音符的判定成功。

9 Near 判定的开关与对应机制

在游戏设置中，我们通常是将 Near 判定关掉的。那么在关闭 Near 判定时，判定机制会发生怎样的变化呢？接下来让我们引入全局开关变量，控制 Near 判定的生效状态。该机制仅影响按歪事件（ E_n^N ）的状态更新，不改变其他判定事件逻辑及按键失效规则。

9.1 开关变量定义

定义全局布尔变量 NearSwitch 表示 Near 判定的启用状态：

$$\text{NearSwitch} \in \{\text{Enabled}, \text{Disabled}\}$$

其中：

- NearSwitch = Enabled: Near 判定正常生效（默认状态，遵循前文第 5.2 节规则）。
- NearSwitch = Disabled: Near 判定关闭，按歪事件的状态更新被抑制。

9.2 关闭状态下的按歪事件修正

当 NearSwitch = Disabled 时，按歪事件 ($E_n^N(I, k)$) 的触发条件保持不变，但事件效果修正为：

$$E_n^N(I, k) \implies \begin{cases} \text{状态保持 } State_n = \text{Active} \text{ (维持活跃)} \\ \text{按键失效 } k \in \text{Inv}(I) \text{ (按键仍进入失效集合)} \end{cases}$$

9.3 特性

1. 抑制状态更新：所有满足原按歪事件触发条件的输入，均不会将音符状态从 Active 更新为 Near，即：

$$\forall n, I, k : (E_n^N(I, k) \wedge \text{NearSwitch} = \text{Disabled}) \implies State_n \not\rightarrow \text{Near}$$

2. 保留失效规则：按键失效机制不受开关状态影响，触发按歪事件的按键仍按第 7 节规则进入失效集合：

$$\forall n, I, k : (E_n^N(I, k) \wedge \text{NearSwitch} = \text{Disabled}) \implies k \in \text{Inv}(I)$$

3. 其他事件独立性：判定成功事件 (E_n^U)、超时事件 (E_n^M) 及无判定事件 (E_n^{None}) 的逻辑不受 NearSwitch 影响，维持原有定义。

9.4 条件约束

1. 开关状态为全局属性，在同一局游戏中保持恒定（初始化后不再动态变更）。
2. 当 NearSwitch = Disabled 时，按歪事件触发后音符消失的视觉效果同样被抑制（音符继续保持活跃状态，直至触发超时事件）。
3. 特殊情况 (PJust 区间重叠音符) 的判定逻辑不受本开关影响，因为该场景仅涉及判定成功事件 (E_n^U)。

10 总结

通过按键输入失效 (4.1/4.2)，失效的范围约束 (6.2)，以及 Near 判定关闭状态下的事件修正 (9.2)，我们就实现了对冗余输入的处理。对于任一音符判定区域，即 $(R_n^U \cup R_n^N)$ 内的全部输入，同一时间帧内所有的冗余输入都会在判定过后失效，而不影响该音符分数的获得。需要注意的是，对冗余输入的处理仅对音符的判定区域 $(R_n^U \cup R_n^N)$ 内有效，超出判定区域的冗余输入仍然会导致误触的发生。