Queuevana Documentation

Queuevana is a family of input-handling utilities designed to make keyboard and mouse input **predictable, ordered, and timing-consistent** under conditions where raw input can become unreliable or hard to manage.

At its core, Queuevana solves one problem:

> **Input events often arrive faster, more chaotically, or less deterministically than applications (especially games) can reliably handle.**

Queuevana introduces a controlled **input queue** between the user and the target application, ensuring clean ordering, optional timing control, and deterministic behavior.

This repository contains documentation for: - **Queuevana** (general-purpose input queue) - **Queuevana Hold** (specialized, niche hold-based blocking variant) - A brief roadmap note on **Queuevana+** (library-oriented future version)

---

1. Queuevana

Overview

Queuevana is a general-purpose input queue system designed primarily for gaming, but usable anywhere consistent input ordering matters.

It captures keyboard and mouse events, places them into a queue, and replays them in a controlled, serialized manner.

This prevents: - Input loss during high APM situations - Inconsistent ordering of key presses/releases - Timing irregularities caused by OS scheduling or application load

Intended Use Cases

Queuevana is suitable for: - Games with tight input timing - Games that drop or mishandle rapid key sequences - High-APM or rhythm-heavy gameplay - Any game using common movement/action keys (WASD, mouse buttons, numbers, etc.)

Queuevana is **not game-specific** and does not assume any particular control scheme.

Key Features

- Deterministic input ordering

- Configurable queue size
- Configurable timing slice and jitter
- Keyboard + mouse support
- Emergency exit hotkey
- Lightweight, low-latency design

How It Works (Conceptual)

1. Input is captured using low-level hooks
2. Each key press/release is enqueued
3. A processor dequeues events one at a time
4. Events are re-sent in strict order with controlled timing

This design decouples **input capture** from **input execution**, which is the core idea behind Queuevana.

Configuration Options

Setting	Description
`SLICE_MS`	Base delay between queued inputs
`JITTER_MS`	Optional random jitter for timing variance
`QUEUE_MAX`	Maximum queued events before overflow protection
`F10`	Emergency exit key

Limitations

- Queuevana does not remap keys
- Queuevana does not automate input
- Queuevana does not inject logic or macros

It strictly **orders and schedules real user input**.

---

2. Queuevana Hold

Overview

Queuevana Hold is a **specialized variant** of Queuevana created for very specific circumstances.

It extends the Queuevana input queue with a **conditional hold-based blocking system**, designed to suppress specific keys *only while another key is held*.

⚠️ **Important:** Queuevana Hold is intentionally niche. It is not meant to replace Queuevana.

Why Queuevana Hold Exists

Queuevana Hold was created to solve a **personal but real input problem**:

- Certain keys (U , P , R) are used extremely frequently during gameplay
- These keys occasionally interfere with other actions when a specific mouse button is held
- Traditional rebinding or macros did not solve the timing or ordering issues

Queuevana Hold solves this by: - Maintaining Queuevana's input queue guarantees - Temporarily blocking specific keys while a hold key (e.g. RMB) is active

---

Intended Use Cases

Queuevana Hold is for: - Very specific control schemes - Highly customized key usage patterns - Situations where *some* keys must be suppressed conditionally

It is **not recommended** for: - General-purpose gaming - Standard WASD-only setups - Users who do not explicitly need conditional blocking

If you do not already know why you need Queuevana Hold, you probably want **Queuevana** instead.

Key Differences from Queuevana

Feature	Queuevana	Queuevana Hold
Input Queue	✔	✔
Timing Control	✔	✔
Conditional Blocking	✖	✔
General-Purpose	✔	✖
Niche / Specialized	✖	✔

How Blocking Works

- A designated **hold key** (default: Right Mouse Button) toggles a held state
- While held:
- Specific keys are suppressed at the hotkey level
- The queue processor enforces an additional safety block
- When released:

- Normal input flow resumes immediately

Blocking is intentionally implemented at **multiple layers** for reliability.

Design Philosophy

Queuevana Hold prioritizes: - Correctness over generality - Predictable behavior over flexibility - Personal workflow optimization

It exists because *one-size-fits-all input handling does not always work*.

---

3. Queuevana+

Status: In Development / Conceptual

Queuevana+ is a planned evolution of Queuevana focused on **library-style usage** rather than standalone gaming scripts.

The goal is to expose Queuevana's input queue system as: - A callable interface - A reusable component - A building block for applications, tools, or accessibility workflows

Intended Use Cases

- Everyday typing workflows
- Application-level input control
- Tooling and automation (non-macro)
- Game development or testing utilities

Queuevana+ is **not limited to gaming**.

---

4. Choosing the Right Version

If you want: - A general solution → **Queuevana** - Conditional hold-based blocking → **Queuevana Hold** - A reusable input system → **Queuevana+ (future)**

---

5. Philosophy & Disclaimer

Queuevana does **not** automate gameplay, cheat, or generate input.

It: - Only processes real user input - Preserves user intent - Improves reliability and predictability

Use responsibly and in accordance with the terms of any software or game you use it with.

6. Closing Notes

Queuevana started as a personal solution to a real input problem and evolved into a robust, reusable system.

Queuevana Hold exists because **sometimes very specific problems deserve very specific tools**.

If that tool fits your workflow — it will feel invisible.

That is the goal.